NAVAL MEDICAL RESEARCH INSTITUTE

Bethesda, MD 20889-5055 NMRI 91-26 March 1991





# COMPUTER-BASED TECHNIQUES FOR COLLECTION OF PULMONARY FUNCTION VARIABLES DURING REST AND EXERCISE

N. A. S. Taylor
J. R. Clarke





Naval Medical Research
and Development Command
Bethesda, Maryland 20889-5044

Department of the Navy
Naval Medical Command
Washington, DC 20372-5210



NOTICES

The opinions and assertions contained herein are the private ones of the writer and are not to be construed as official or reflecting the views of the naval service at large.

When U. S. Government drawings, specifications, or other data are used for any purpose other than a definitely related Government procurement operation, the Government thereby incurs no responsibility nor any obligation whatsoever, and the fact that the Government may have formulated, furnished or in any way supplied the said drawings, specifications, or other data is not to be regarded by implication or otherwise, as in any manner licensing the holder or any other person or corporation, or conveying any rights or permission to manufacture, use, or sell any patented invention that may in any way be related thereto.

Please do not request copies of this report from the Naval Medical Research Institute. Additional copies may be purchased from:

National Technical Information Service
5285 Port Royal Road
Springfield, Virginia 22161

Federal Government agencies and their contractors registered with the Defense Technical Information Center should direct requests for copies of this report to:

Defense Technical Information Center
Cameron Station
Alexandria, Virginia 22304-6145

## TECHNICAL REVIEW AND APPROVAL

### NMRI 91-26

The experiments reported herein were conducted according to the principles set forth in the current edition of the "Guide for the Care and Use of Laboratory Animals," Institute of Laboratory Animal Resources, National Research Council.

This technical report has been reviewed by the NMRI scientific and public affairs staff and is approved for publication. It is releasable to the National Technical Information Service where it will be available to the general public, including foreign nations.

LARRY W. LAUGHLIN
CAPT, MC, USN
Commanding Officer
Naval Medical Research Institute

# REPORT DOCUMENTATION PAGE

| | |
|---|---|
| 1a. REPORT SECURITY CLASSIFICATION<br>UNCL | 1b. RESTRICTIVE MARKINGS |
| 2a. SECURITY CLASSIFICATION AUTHORITY | 3. DISTRIBUTION/AVAILABILITY OF REPORT<br>Approved for public release; distribution is unlimited |
| 2b. DECLASSIFICATION/DOWNGRADING SCHEDULE | |
| 4. PERFORMING ORGANIZATION REPORT NUMBER(S)<br>NMRI 91-26 | 5. MONITORING ORGANIZATION REPORT NUMBER(S) |
| 6a. NAME OF PERFORMING ORGANIZATION<br>Naval Medical Research Institute<br>6b. OFFICE SYMBOL (If applicable) | 7a. NAME OF MONITORING ORGANIZATION<br>Naval Medical Command |
| 6c. ADDRESS (City, State, and ZIP Code)<br>8901 Wisconsin Avenue<br>Bethesda, MD 20814-5055 | 7b. ADDRESS (City, State, and ZIP Code)<br>Department of the Navy<br>Washington, DC 20372-5120 |
| 8a. NAME OF FUNDING/SPONSORING ORGANIZATION Naval Medical Research & Development Comman<br>8b. OFFICE SYMBOL (If applicable) | 9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER |
| 8c. ADDRESS (City, State, and ZIP Code)<br>8901 Wisconsin Avenue<br>Bethesda, MD 20814-5044 | 10. SOURCE OF FUNDING NUMBERS<br>PROGRAM ELEMENT NO. 63713N; PROJECT NO. M0099; TASK NO. .01B; WORK UNIT ACCESSION NO. 1005 |
| 11. TITLE (Include Security Classification)<br>(U) COMPUTER-BASED TECHNIQUES FOR COLLECTION OF PULMONARY FUNCTION VARIABLES DURING REST AND EXERCISE | |
| 12. PERSONAL AUTHOR(S)<br>Nigel A.S. Taylor and John R. Clarke | |
| 13a. TYPE OF REPORT<br>Technical Report<br>13b. TIME COVERED FROM 01/90 TO 12/90 | 14. DATE OF REPORT (Year, Month, Day)<br>1991 March<br>15. PAGE COUNT<br>76 |
| 16. SUPPLEMENTARY NOTATION | |

| 17. COSATI CODES | | | 18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number) |
|---|---|---|---|
| FIELD | GROUP | SUB-GROUP | lung volumes, flow-volume loops, respiratory timing, $FEV_{1.0}$, maximal voluntary ventilation, post-inspiratory and post-expiratory pauses, peak expiratory flow |
| | | | |
| | | | |

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

Pulmonary function tests are routinely included as part of the experimental protocol for both immersion and barophysiology experiments. Typically, these tests are performed using standard spirometry, and as such suffer from time consuming data analysis and experimental error associated with such analysis. In many cases the combined effects of these limitations has meant that some valuable data often fails to come out in the analysis. One solution to this situation is to transfer these tasks to the computer. We have developed a computer-based system for both data collection and analysis, which is faster, more sensitive to minor physiological perturbations, and more precise than standard spirometry. The apparatus and programming algorithms for this system are presented in this report so that other laboratories might be able to similarly improve the methods for collection of pulmonary function data.

| | | |
|---|---|---|
| 20. DISTRIBUTION/AVAILABILITY OF ABSTRACT<br>☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT ☐ DTIC USERS | 21. ABSTRACT SECURITY CLASSIFICATION<br>Unclassified | |
| 22a. NAME OF RESPONSIBLE INDIVIDUAL<br>Regina E. Hunt, Command Editor | 22b. TELEPHONE (Include Area Code)<br>(202) 295-0198 | 22c. OFFICE SYMBOL<br>SD/RSD/NMRI |

## TABLE OF CONTENTS


ABSTRACT . . . . . i

ACKNOWLEDGEMENTS . . . . . iv

INTRODUCTION . . . . . 1

METHODS . . . . . 3

- Apparatus . . . . . 3
- Data acquisition . . . . . 4
- Calibration . . . . . 6
- Rest and exercise respiratory patterns . . . . . 7
- Lung volume measures . . . . . 7
- Maximal voluntary ventilation . . . . . 10
- Flow-volume loops and derived variables . . . . . 10

REFERENCES . . . . . 12

FIGURE 1 . . . . . 9

FIGURE 2 . . . . . 11

APPENDIX 1 . . . . . 13

APPENDIX 2 . . . . . 19

- Part I . . . . . 19
- Part II . . . . . 31

APPENDIX 3 . . . . . 34

APPENDIX 4 . . . . . 39

APPENDIX 5 . . . . . 40

APPENDIX 6 . . . . . 41

APPENDIX 7 . . . . . 44

APPENDIX 8 . . . . . 57

GLOSSARY OF TERMS AND ABBREVIATIONS . . . . . 71

## ACKNOWLEDGEMENTS

Naval Medical Research and Development Command, Research Task No. M0099.01B.1005. This work was completed while the author held a National Research Council Research Associateship at the Naval Medical Research Institute. The views, opinions and/or observations should not be construed as or reflecting the views, policy, or decisions of the Navy Department or the naval service at large, unless so designated by other official documentation.

## INTRODUCTION

Pulmonary Function Tests (PFTs) are routinely included in experimental protocols involving hyper- and hypobaric excursions. Unfortunately, the full potential of those tests is often not realized due to the limited capabilities of conventional data recording technology. A potential improvement came in the last decade as microcomputers became available for clinical PFTs. However, microcomputer use in the reseach setting has been limited due to the perceived "black box" (i.e. undocumented) nature of many such devices. Nevertheless, the American Thoracic Society (ATS) stated the advantages of computer acquisition of pulmonary function data (1) as follows:

"(1) Complete automation of a procedure may result in significantly reduced time and cost, and in increased accuracy;

(2) Assurance that standardized procedures are followed;

(3) Significant reduction in major measurement errors;

(4) Storage and retrieval of information quickly and efficiently;

(5) Implementation of automated calibration and system check procedures within the instrument;

(6) Standardized and consistent interpretation of results."

A recent example illustrates the power in these statements. Respiratory flow data acquired by a computer on a dive to 450 m revealed oscillations that were suspiciously similar to the frequency signature of HPNS tremor (2). Had a conventional mechanical spirometer been used, the oscillations might have been dismissed as mechanical artifacts. However, the accuracy and frequency resolution of the computer lent credence to the hypothesis that the observed oscillations represented a previously unknown action of the High Pressure Nervous Syndrome.

The ATS suggests comprehensive quality assurance guidelines for the use of digital computers in the laboratory (3). Those guidelines were refered to in the ATS standards for spirometry promulgated in 1987 (4). According to the standards, users of PFT software should possess (and hopefully understand) source code listings (computer instructions written in a high level language such as Basic, C, or Fortran). Unfortunately, commercial software is typically proprietary, meaning source code is not

available. Even when code is presented, it is frequently poorly documented. These restrictions are undesirable in the clinical setting, but are intolerable in scientific and engineering tasks.

The purpose of this report is to describe the hardware installation and source code for a Pulmonary Function data acquisition system that has proven useful in the hyperbaric research laboratory. It illustrates how computers can economically collect, analyze and display data from a variety of pulmonary function tests (Table 1). [Readers interested in computer aided measurement of dynamic, quasi-static and static pulmonary compliance are directed to Taylor (5).]

## METHODS

### Apparatus

The current laboratory uses a PC(AT) clone (Model Z-248, Zenith Data Systems, St. Joseph, MI) although any IBM compatible system would suffice. The critical component is the analogue-to-digital (A/D) convertor, as this determines both the maximal sampling rate and the ease of software development. The MetraByte DAS-16F A/D board (Metrabyte Corp., Taunton, MA) has a maximal sampling rate of 100 kHz, can take single ended input from 16 channels (or eight differential analogue inputs), permits both foreground and background data collection, and is fully programmable using either commercially available or laboratory-developed software. Flow was measured by a Hans Rudolph pneumotachograph (Model 4813, Hans Rudolph, Inc., Kansas City, MO), heated to 37 °C and coupled to a Validyne differential pressure transducer (MP45 ± 0.196 kPa (±2 $cmH_2O$), Validyne Engineering, Northridge, CA). Mouth pressure was obtained using a second Validyne differential pressure transducer (MP45 ± 9.803 kPa (±100 $cmH_2O$)) connected to the airway between the mouth and the pneumotachograph. Unfiltered output from the flow and pressure transducers was amplified using Validyne CD18 and CD19 carrier demodulators, respectively.

The frequency content of flow generated during the Forced Vital Capacity maneuver extends to at least 12 Hz (6). Digital sampling theory (7) requires sampling to occur at frequencies at least double that in the sampled signal. Accordingly, data from both inputs was sampled at 50 Hz. Sampling at even 100 Hz is not uncommon.

[Validyne is now manufacturing an interface card (UPC607) which serves both data acquisition and signal conditioning needs, removing the need for separate signal demodulators].

Before the DAS-16F board can be used it must first be correctly installed into the computer. This installation involves both hardware and software. Physical installation requires the setting of four D.I.P. switches, and then the correct positioning of the board. The user must correctly position the switches for: (a) channel configuration (16 or 8 channel use); (b) unipolar (positive only) or bipolar (positive and negative) voltage inputs; (c) direct memory access level (select 1 for systems with hard-disk and 3 for

floppy drive systems); and (d) gain, which operates interactively with switch (b) to set the input voltage range. Software installation is performed using INSTALL.EXE (software provided with the board), which sets the desired base address for the board within your system. Once installed the card should be calibrated using supplied software (CALF.EXE or CAL.BAS; see DAS-16F Manual).

Data acquisition

Appendix 1 contains a generic subroutine for the collection of data in either foreground mode (all other computer functions temporarily suspended) or background mode (for simultaneous data collection and processing). Data acquisition subroutines are presented before calibration subroutines, since to calibrate using the computer you must first be able to collect data. All computer programs were written using Microsoft Fortran77, and are presented in their smallest parts (subroutines).

The information below provides a brief overview of the commands issued by the Fortran software to the MetraByte board. Before data collection, four initializations (operation modes) must be completed:

(1) Mode 0: This mode is used to initialize the A/D board by first testing its physical presence, then setting an input/output (base) address (variable 1), setting the interrupt level (variable 2), and setting the direct memory access (D.M.A.) level (variable 3). The D.M.A. level is set both physically on the board and within the software (see DAS-16F Manual).

(2) Mode 1: This mode sets the lower (variable 1) and upper ranges (variable 2) of the channels to be scanned.

(3) Mode 17: This mode sets the programmable timer rate and therefore determines the sampling rate, and requires two variables: (a) divide data for counter 2 (variable 1), and (b) divide data for counter 1 (variable 2). These counters of the 8254 counter timer produce the output pulse rate for triggering the A/D conversions. The sampling frequency is derived from the following equation:

Frequency = 1,000,000/[(variable 1/number of channels)* variable 2] However, the user is responsible for doing this division (a point not in the manual), so set the size of variables 1 and 2 with this in mind. For example, to set a sampling rate of 50 Hz for 1 channel variable 1 is set to 1000 and variable 2 is set to 20. To collect 2 channels at 50 Hz, leave variable 2 at 20 and set variable 1 to 500 (i.e. 1000/2).

(4) Mode 19: This last initialization mode allows for the use of an external trigger (positive input voltage) to initiate the start of A/D conversions. This is an extremely useful feature which permits remote operation by either the experimenter or the subject. It effectively suspends the computer program until a pre-determined input voltage is received. Three variables are required to operate this mode:

(a) channel number for the analogue input voltage,

(b) the magnitude of the trigger voltage (bit equivalent), and

(c) the slope of the input signal (i.e. positive or negative).

Data acquisition is performed using one of several modes (see DAS-16F Manual), however Modes 4 (foreground) and 5 (background) are sufficient for most applications:

(5) Mode 4: Performs A/D conversions in the foreground according to the channels ranges and sampling rates determined above. Three variables values are required: (a) the number of conversions (word count) to be performed (e.g. for 2 channels collecting data at 50 Hz for 10 seconds, word count = 2 * 50 * 10); (b) an array (previously initialized and dimensioned within the Fortran code) into which the input shall be loaded, and the position within the array from which to start the storage (this is usually zero for a single

dimension array or '[1,0]' for an array with several levels); and (c) the source of the trigger to start the A/D conversions (set this to 1 to use the previously programmed interval timer, Mode 17).

(6) Mode 5: Performs A/D conversions in the background according to the channels ranges and sampling rates determined above. Four input variables are required: (a) the number of conversions (word count) to be performed (as above); (b) the segment (16 bit) of the memory to receive data (you need to seek technical help here as this location is dependent upon the memory allocations already assigned within your system; we use #5000 and #6000 {Fortran}, or &H5000 and &H6000 {Basic}); (c) the source of the trigger input to start the A/D conversions (as above); and (d) a command telling the computer to carry out this sequence once (single cycle) or to operate continuously (until stopped using Mode 7 - see DAS-16F Manual).

Calibration

Calibration of the pneumotachograph must be performed under two conditions; zero flow, and using a standard volume (e.g. 3 liter) passed through the pneumotachograph at flow rates appropriate to a given experiment. Using an analysis routine, flow is integrated with respect to time, providing both volume and flow calibration. This method is superior to electronic integration, since the latter technique invariably produces an electronic drift in the baseline, making data interpretation impossible.

Mouth pressure is often of interest, especially during exercise protocols. Prior to calibration the linearity of the mouth pressure transducer must be verified. Subsequent calibration may then be performed using a two point calibration over the positive range of anticipated mouth pressures. From linear regression, the pressure data yielded both the zero offset and slope for the calibration. These

values were saved and used during data analysis. The calibration subroutines for the pneumotachograph and mouth pressure transducer are contained in Appendix 2.

Rest and exercise respiratory patterns

Respiratory pattern data (which includes $V_T$, $V_E$, and respiratory timing variables) were derived from the same routines that provided lung volume data. The difference lay in the breathing pattern used by the subject, rather than within the computer program. The example provided in Appendix 3 is from a program written to analyze data collected at 450 m (changes in ambient pressure are automatically taken into account by these procedures). Subjects exercised for 25 min while exposed to a moderate external airway resistance. Appendix 8 contains the entire program. This program also included the measurement of mouth pressure (including the determination of peak-to-peak values and inspiratory to expiratory ratios) and the integration of mouth pressure into trajectory data for a flow-volume-pressure three-dimensional surface (8).

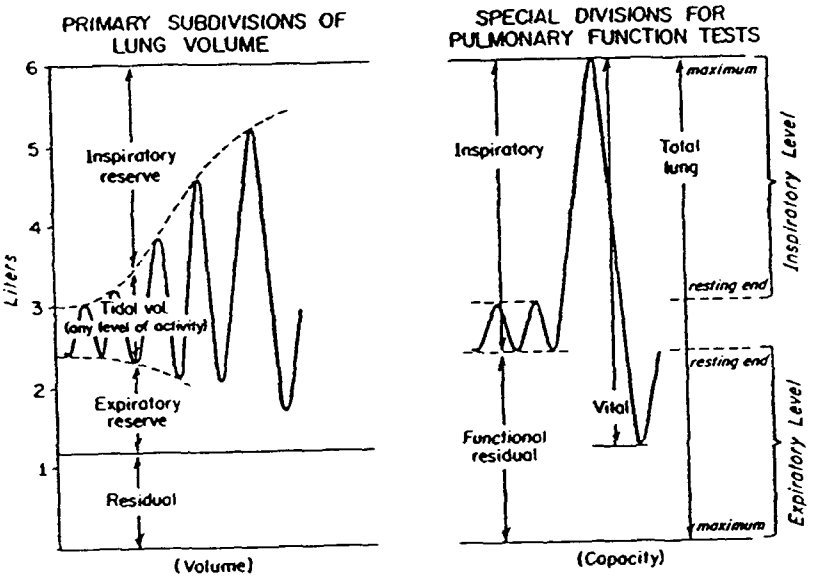
Lung volume measures

The measurement of lung volumes (excluding RV) relies more upon the skills of the experimenter than the art of the programmer. Subjects must be trained to provide reproducible VC maneuvers through the attainment of consistent RV and TLC end points. Once this is accomplished the experimenter can, with a single maneuver, obtain VC, ERV, IC, and IRV (Figure 1). Following an exhalation to RV, the subject slowly inspired to maximum lung capacity (TLC), followed by an expiration to RV and a normal slow inspiration. The subject then took normal breaths (giving $V_T$, ERV, and by subtraction, IC). From this point the subject inspired to TLC again. By subtracting from IC the $V_T$ obtained using Appendix 3 (since the $V_T$ in this maneuver is often falsely large), IRV is derived.

Subroutines necessary for measuring the above lung volumes are contained in Appendix 4. Note that the programming code is the same for each lung volume; the changes being only in the maneuver performed by the subject. These subroutines, and those included below (except the flow-volume subroutines), perform analysis on complete respiratory cycles only. This is essential, since during exercise subjects do not commence inspiration at the same time as the computer starts to collect data,

and rarely terminate exhalation immediately prior to the end of data collection. Therefore, within the initial analysis subroutine (Appendix 3) is a 'paragraph' which searches the flow data to find the first commencement of inspiration. This is found when the flow data crosses the predetermined basal flow threshold in the inspiratory direction. The subroutine then counts the number of inspirations (counter I) and expirations (counter E). To ensure these are always equal, the last inspiration is omitted, and the output is always written for I-1 inspirations and expirations.

# FIGURE 1




Standard lung volumes and capacities.

Maximal voluntary ventilation

The MVV maneuver (Appendix 5) is also a hybrid of the breathing pattern subroutines (Appendix 3). For the MVV, all the $V_T$s are summed then divided by the time taken to complete the maneuver. This time is controlled by the experimenter so that it equals the duration of interest (i.e. 12 or 15 seconds). As with the above subroutines, only complete cycles are analyzed, so the time to complete the maneuver (as determined by the software) will often not equal the time used during the experiment. This difference is minimal, and results in greater precision in the determination of MVV than is possible using a spirometer.

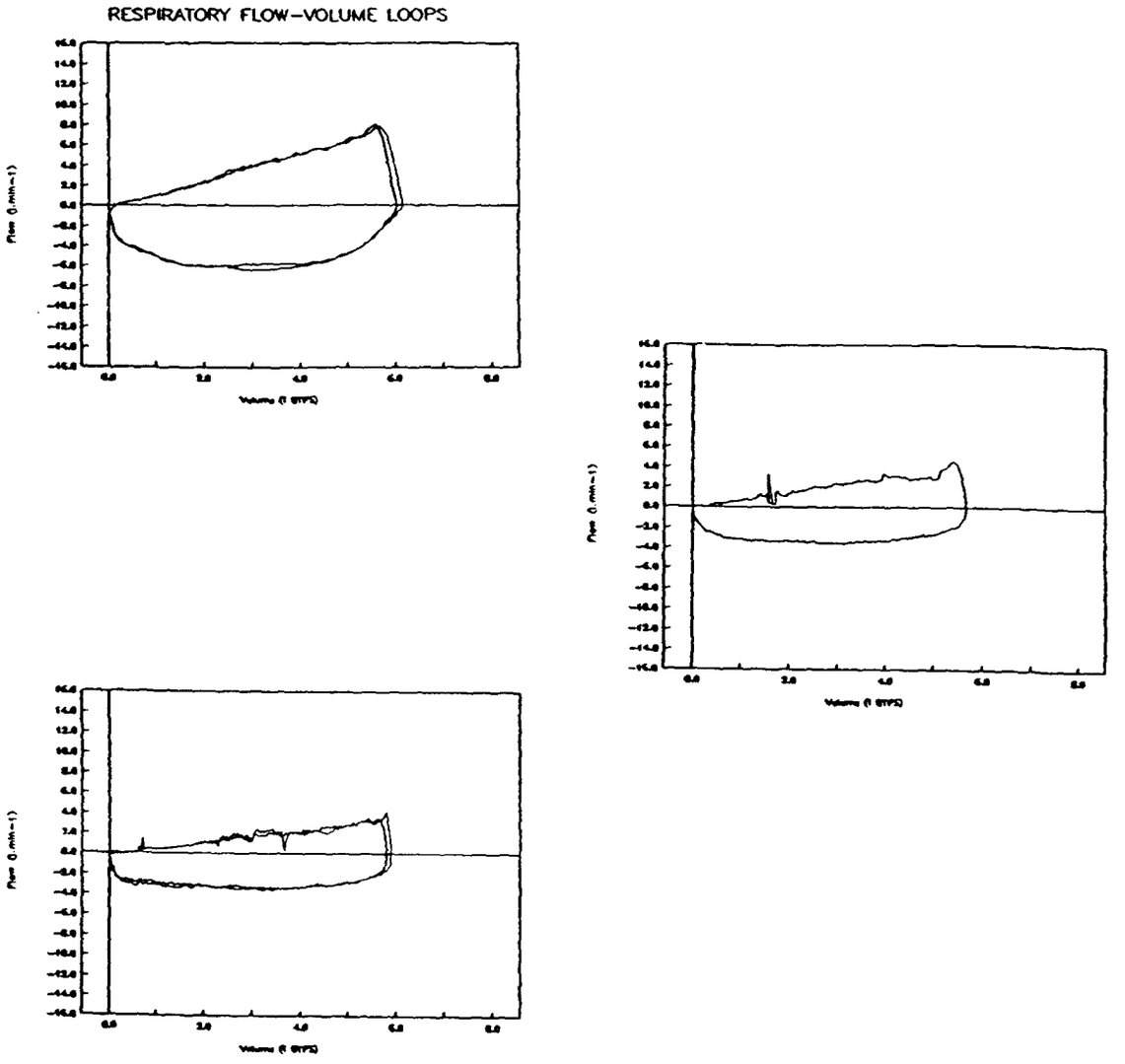
Flow-volume loops and derived variables

For this maneuver (Appendix 6) subject training was again vital to ensure that subjects obtained true maximal flows and reproducible end points at RV and TLC. The amplifier gain setting was adjusted so that voltages during peak flows were as close as possible to the $\pm$5 Volt input capacity of the A/D board.

Since data sampling was at 50 Hz, flow and instantaneous lung volumes were derived at 20 ms intervals over the entire volume excursion. These data were saved as an ASCII file for importation into suitable graphics packages (e.g. Lotus123 (Figure 2) or Sigmaplot [see: Taylor 1990]).

The flow data obtained from these maneuvers was also used to obtain the $FEV_{1.0}$ for each expiration. The ASCII flow data was searched manually to detect PEFR and MEFR for various percentages of the VC. While algorithms for these latter computations may be written and incorporated into the analysis subroutines, it is often easier to take these values by hand directly from the ASCII files. If noise occured in the flow signal at the volume of interest, computer analysis could be confounded.

# FIGURE 2




Flow-volume loops obtained during trials conducted at the surface, 500 fsw and at 1000 fsw. All plots are for the same subject.

## REFERENCES


1. Gardner RM, Clausen JL, Cotton DJ, Crapo RO, Epler GR, Hankinson JL, and Johnson RL, Jr. Computer guidelines for pulmonary laboratories. American Review of Respiratory Disease, Vol. 134, No. 3, pp. 628-629, 1986.
2. Fraser I.M., and Flook V., "HPNS tremor on flow-volume curves at 450 meters." In: Sterk W. and Geeraedts L. (eds.). Proceedings of the European Undersea Biomedical Research Society. Amsterdam, Netherlands. August 11-18, 1990, pp. 75-81.
3. ATS Statement. Quality assurance in pulmonary function laboratories. American Review of Respiratory Disease, Vol. 134, pp. 625-627, 1986.
4. Gardner RM, Hankinson JL, Clausen JL, Crapo RO, Johnson RL, Jr, and Epler GR. Standardization of spirometry - 1987 update. American Review of Respiratory Disease, Vol 136, pp. 1285-1298, 1987.
5. Taylor N.A.S., "Computer integrated measurement of respiratory compliance." Computers in Biology and Medicine, Vol. 20, No. 1, pp. 55-64, 1990.
6. Lemen RJ, Gerdes CB, Wegmann MJ, and Perrin KJ. Frequency spectra of flow and volume events for forced vital capacity. Journal of Applied Physiology, Vol. 53, pp. 977-984, 1982.
7. Jerri AJ. The Shannon sampling theorem-its various extensions and applications; a tutorial review. Proceedings of the IEEE, Vol. 65, pp. 1565-1596, 1977.
8. Marshall RN, Mazur SM, and Taylor NAS. Three dimensional surfaces for human muscle kinetics. European Journal of Applied Physiology, 1990, In press.

APPENDIX 1

Data acquisition subroutines.

```
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
c            GENERIC DATA COLLECTION PROGRAM
c
c   Program uses Metrabyte A/D conversion board, collects data
c   at 50 Hz from channels 3 and 4. Collection occurs in the background
c   (Mode 5) and takes place in 1 minute segments. Between each segment
c   the program looks at channel 3 for analog input from an external
c   trigger. This input terminates the data collection. Data is continuously
c   written to disk in ASCII code between successive samplings.
c
c    NIGEL A.S. TAYLOR                    4th MAY 1990
c    DIVER LIFE SUPPORT EQUIPMENT PROGRAM, NAVAL MEDICAL RESEARCH INSTITUTE
c
c    LATEST UPDATE: 5/24/90
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
$STORAGE:2
      INTEGER PARAM,IMODE,OFFADR,DASH16,RCODE,J,I,L,M,N,O,ZZ
      INTEGER DATA1,DATA2,LMODE,LCODE,LPARAM,JCODE,JMODE,COUNT
      INTEGER ANS1,ANS2,TIME,CHECK,KCODE,KMODE,VAR,ELEM,TCOUNT
      DIMENSION DATA1(3,3001),PARAM(4),DATA2(3,3001)
      DIMENSION LPARAM(5),ELEM(5),VAR(5)
      CHARACTER*20 FILENAME
      CHARACTER BELL
      BELL = CHAR(7)
c
c  The bell is used to produce a 'beep' from the PC. This prompts
c  the experimenter for input. See the 'WRITE' statements below.
c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
      OPEN(*)
100   WRITE(*,1)
1     FORMAT(//////////
     *    '          ENTER NAME OF OUTPUT FILE FOR DATA:'/////////)
      READ(*,11) FILENAME
11    FORMAT(A20)
      OPEN(3,FILE=FILENAME,STATUS='NEW',FORM='FORMATTED')
      WRITE(*,7000)
7000  FORMAT(//' ENTER EXPERIMENTAL DURATION IN MINUTES (I3):'//)
      READ(*,7001) TIME
7001  FORMAT(I6)
      TCOUNT=0
      CHECK=0
      ZZ=0
c
```

```
c
c Initialization of the MetraByte A/D board: see text for description
c
      IMODE = 0
      PARAM(1) = #300
      PARAM(2) = 2
      PARAM(3) = 3
      RCODE = DASH16(IMODE, PARAM)
c
c Setting the channel scanning limits for 3 channels, starting at 2 and
c ending at 4.
c
      IMODE = 1
      PARAM(1) = 2
      PARAM(2) = 4
      RCODE = DASH16(IMODE, PARAM)
c
c Setting the sampling rate for three channels at 50 Hz
c (i.e. 1,000,000/[{333=1000/3}*20])

      IMODE = 17
      PARAM(1) = 333
      PARAM(2) = 20
      RCODE = DASH16(IMODE, PARAM)
      WRITE(*,12)
12    FORMAT(///'              PRESS THE TRIGGER TO'/
     *  '    ****** ACTIVATE DATA COLLECTION ******'/////)
c
c  Using an analog trigger on channel 2 to activate data collection.
c  Program pauses until an positive voltage greater than 300 (binary
c  units) is detected. This is a very useful Mode that can be used at
c  any point in the program. Unlike large machines, the PC is unable
c  to check hardware status while completing other tasks, so this Mode
c  can also be used to terminate operations as well as to initiate.
c
      IMODE = 19
      PARAM(1) = 2
      PARAM(2) = 300
      PARAM(3) = 0
      RCODE = DASH16(IMODE, PARAM)
c
      WRITE(*,141) BELL
      WRITE(*,13)
13    FORMAT(////
     *  '    **********************************************'/
     *  '    *** C O L L E C T I O N  S T A R T E D ***'/
     *  '    **********************************************'///)
c
c Background data collection using Mode 5. See test for description.
c Note that the Variable/Arrays for this CALL have been altered from
c previous modes. This is because they need to be unique to this CALL
c since it will be operating in the background and will be confused if
```

```
c these values are altered during operation.
c
        LMODE = 5
        LPARAM(1) = 9000
        LPARAM(2) = #5000
        LPARAM(3) = 1
        LPARAM(4) = 0
        LCODE = DASH16(LMODE, LPARAM)
        ZZ = ZZ + 1
c
7777    CONTINUE
c

c
c Mode 8 is used to monitor the status of the background data collection.
c In this example we wish 1 full minute of data collection to be transferred
c into an array from memory (3 chans * 50 Hz * 60 sec = 9000). Mode 8 keeps
c the program sitting till the 'word count' equals 9000. We could use this
c Mode to transfer data in smaller segments for real-time processing. This is
c the real advantage that Mode 5 provides.
c
3330    JMODE = 8
        JCODE = DASH16(JMODE, VAR)
        IF (VAR(3).GE.9000) GOTO 3001
        GOTO 3330
3001    WRITE(*,3900) ZZ
3900    FORMAT('+       MINUTE ',I2,' COMPLETE:')
c
c Here Mode 4 is used to monitor channel 2 for a positive input from
c the trigger (i.e. greater than binary value of 500). Mode 1 is not
c re-defined, instead all 3 channels are monitored. However, the 'word
c count' is set to 1 to remove the input from the other channels.
c Note: this only works because the trigger is connected to channel 2.
c If trigger input is detected the program jumps to Statement 4000 to
c save the data, then to Statement 8 to terminate data collection.
c
        IMODE = 4
        PARAM(1) = 1
        PARAM(2) = OFFADR(CHECK)
        PARAM(3) = 1
        RCODE = DASH16(IMODE, PARAM)
        IF (CHECK.GT.500) GOTO 4000
        IF (TCOUNT.GE.TIME) GOTO 4000


c Mode 5 is activated again immediately after the trigger check. This
c results in minimal data loss (milliseconds).
c
        LMODE = 5
        LPARAM(1) = 9000
```

```
        LPARAM(2) = #6000
        LPARAM(3) = 1
        LPARAM(4) = 0
        LCODE = DASH16(LMODE, LPARAM)
        ZZ = ZZ + 1
c
c Mode 9 is then used to transfer data from memory to a previously
c initialized and dimensioned array. This occurs in the foreground
c while the data acquisition subroutines are operating in background.
c Five input Variables are required for Mode 9:
c (1) 'word count', 9000 = 3 chans * 50 Hz * 60 seconds
c (2) memory address as per Mode 5 declaration
c (3) the conversion number to start the transfer from
c (4) the array name and location within the array (e.g. point 0 of level 1)
c (5) channel array data: use 0 if not required.
c
4000    KMODE = 9
        ELEM(1) = 9000
        ELEM(2) = #5000
        ELEM(3) = 0
        ELEM(4) = OFFADR(DATA1(1,0))
        ELEM(5) = 0
        KCODE = DASH16(KMODE, ELEM)
c
c Now while Mode 5 is still operating the data are written to a
c file on the disk in ASCII code. This file is not closed, but is
c continually added to with data from each usuccessive minute.
c
        DO 3002 I=1,(9000/3)
        WRITE(3,4949) (DATA1(J,I),J=2,3)
3002    CONTINUE
4949    FORMAT(2I6)
c
        TCOUNT = TCOUNT + 1
        IF (CHECK.GT.500) GOTO 8
        IF (TCOUNT.GE.TIME) GOTO 8
c
3331    JMODE = 8
        JCODE = DASH16(JMODE, VAR)
        IF (VAR(3).GE.9000) GOTO 3004
        GOTO 3331
3004    WRITE(*,3900) ZZ
c
        IMODE = 4
        PARAM(1) = 1
        PARAM(2) = OFFADR(CHECK)
        PARAM(3) = 1
        RCODE = DASH16(IMODE, PARAM)
c
        IF (CHECK.GT.500) GOTO 4001
        IF (TCOUNT.GE.TIME) GOTO 4001
c
```

```
      LMODE = 5
      LPARAM(1) = 9000
      LPARAM(2) = #5000
      LPARAM(3) = 1
      LPARAM(4) = 0
      LCODE = DASH16(LMODE, LPARAM)
      ZZ = ZZ + 1
c
4001  KMODE = 9
      ELEM(1) = 9000
      ELEM(2) = #6000
      ELEM(3) = 0
      ELEM(4) = OFFADR(DATA2(1,0))
      ELEM(5) = 0
      KCODE = DASH16(KMODE, ELEM)
c
      DO 3005 I=1,(9000/3)
      WRITE(3,4949) (DATA2(J,I),J=2,3)
3005  CONTINUE
c
      TCOUNT = TCOUNT + 1
      IF (CHECK.GT.500) GOTO 8
      IF (TCOUNT.GE.TIME) GOTO 8
c
      GOTO 7777
c
8     WRITE(*,141) BELL,BELL,BELL,BELL,BELL
141   FORMAT(' ',A1,' ',A1,' ',A1,' ',A1,' ',A1)
      WRITE(*,14)
14    FORMAT(//
     *  '   **********************************************'/
     *  '   *** C O L L E C T I O N  F I N I S H E D ***'/
     *  '   **********************************************'/)
20    FORMAT(I1)
c
      CLOSE(3)
c




      WRITE(*,2101) FILENAME
2101  FORMAT(//////////
     *  '        DATA FROM: ',A20,' SAVED:'///
     *  '   **********************************************'/
     *  '   R E C O R D  E X P E R I M E N T  T I M E ! !'/
     *  '   **********************************************'//
     *  '        WHAT DO YOU WISH TO DO NEXT?'/
     *  '        1 RETURN = COLLECT MORE DATA'/
     *  '        9 RETURN = TERMINATE PROGRAM'/)
      READ(*,20) ANS2
      IF (ANS2.EQ.1) GOTO 100
```

```
      IF (ANS2.EQ.9) CONTINUE
C
9999  WRITE(*,9191)
9191  FORMAT(//////////////
     *   ' **********************************************'/
     *   ' ********    G O O D    B Y E    **********'/
     *   ' **********************************************'/////)
      STOP
      END
CCCC
```

## APPENDIX 2

Calibration subroutines.

PART I: Collecting the calibration data.

```
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
c            EXAMPLE CALIBRATION Program:
c
c   Refer to the GENERIC DATA COLLECTION program for explanations
c   of various programming CALL routines and functions.
c
c      CALIBRATION FOR MOUTH PRESSURE, END-TIDAL CO2 AND
c                  PNEUMOTACHOGRAPH
c
c         SET-UP FOR THE DASH16 A/D BOARD (MetraByte)
c    Data collection started from input to channel two.
c    Sampling at 50Hz.
c    Four channels monitored: 2=trigger, 3=mouth pressure
c                             4=flow, 5=end-tidal CO2.
c
c    NIGEL A.S. TAYLOR                  May 1990
c    Diving Medicine, Naval Medical Research Institute.
c
c    LATEST UPDATE: 21 MAY 1990
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc

$STORAGE:2
        INTEGER ZERO,GAIN,PEAKMODE,ZCHAN,HIGH,MM,M,B,D,E,F,G,H,J,K,IMODE
        INTEGER ZER,HIG,COUNT,GAIN2,BASE,BAS,ICHAN,BIG,BOG,PARAM,SAVE
        INTEGER MEANNOFLOW,LITER,NOFLOW,GAIN3,ANS,DASH16,OFFADR,RCODE
        REAL MEANZERO,MEANHIGH,PRESSURE,REGCOEFF,CONSTANT
        REAL V,W,X,Y,Z
        REAL RZERO,RHIGH,A,C,RBASE,RBIG,PRESS2
        CHARACTER*40 FILENAME,FILE2,FILE3,FILE4
        CHARACTER BELL
        DIMENSION ICHAN(2),ZERO(11),HIGH(11),RZERO(11),RHIGH(11)
        DIMENSION BASE(11),RBASE(11),BIG(11),RBIG(11),PARAM(4)
        DIMENSION NOFLOW(101),LITER(2000)
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
        OPEN(*)
        BELL = CHAR(7)
100     WRITE(*,1)
1       FORMAT(////////////////
     *  '  PRESSURE, PetCO2 AND VOLUME CALIBRATION Program.'//////
     *  '      NOTE: YOUR FIRST CALIBRATION MUST BE THE'/
     *  '       TRANSDUCER MEASURING MOUTH PRESSURE.'//
     *  '     ENTER 1 [RETURN] TO COMMENCE CALIBRATION'///)
        READ(*,20) ANS
```

```
      IF (ANS.EQ.1) CONTINUE
20    FORMAT(I1)
c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c     Pressure calibration
c
10    IMODE = 0
      PARAM(1) = #300
      PARAM(2) = 2
      PARAM(3) = 3
      RCODE = DASH16(IMODE, PARAM)
c
      IMODE = 1
      PARAM(1) = 3
      PARAM(2) = 3
      RCODE = DASH16(IMODE, PARAM)
c
      IMODE = 17
      PARAM(1) = 1000
      PARAM(2) = 20
      RCODE = DASH16(IMODE, PARAM)
c
      WRITE(*,3)
3     FORMAT(///'   SET THE PRESSURE TRANSDUCER AT ZERO PRESSURE.'/
     *  '       IS TRIGGER CONNECTED TO CHANNEL TWO?'/
     *  '     IS TRANSDUCER CONNECTED TO CHANNEL THREE?'/////
     *  '      ***** PRESS TRIGGER TO COMMENCE *****'//)
c
      IMODE = 19
      PARAM(1) = 2
      PARAM(2) = 300
      PARAM(3) = 0
      RCODE = DASH16(IMODE, PARAM)
c
      WRITE(*,31)
31    FORMAT(/' ****** COLLECTION HAS BEGUN ******'/)
      WRITE(*,4000) BELL
c
      IMODE = 4
      PARAM(1) = 12
      PARAM(2) = OFFADR(ZERO(0))
      PARAM(3) = 1
      RCODE = DASH16(IMODE, PARAM)
c
c Summing 10 zero pressure points to obtain the mean (below).
c
      A = 0.0
      DO 311 J=2,11
      ZER=ZERO(J)
      RZERO(J)=FLOAT(ZER)
      A = A + RZERO(J)
```

```
311     CONTINUE
c
        WRITE(*,4)
4       FORMAT(/' ****** CALIBRATION VALUES COLLECTED ******'/)
        WRITE(*,4000) BELL
c
c  Calculating mean zero pressure.
c
        MEANZERO = A / 10.0
        WRITE(*,41) (RZERO(J),J=2,11),MEANZERO
41      FORMAT(2(5(2X,F9.2)//),' MEAN ZERO PRESSURE = ',F9.2//)
        WRITE(*,42)
42      FORMAT(' CHECK BASAL VALUES ARE ALL ABOUT EQUAL.')
c
        WRITE(*,43)
43      FORMAT(//' WHAT DO YOU WISH TO DO NEXT?'/
     *  ' 1 RETURN = CONTINUE TO NEXT STAGE,'/
     *  ' 6 RETURN = REPEAT & ERASE BASAL CALIBRATION,'/
     *  ' 9 RETURN = END PROGRAM.'/)
        READ(*,20) B
        IF (B.EQ.1) CONTINUE
        IF (B.EQ.6) GOTO 10
        IF (B.EQ.9) GOTO 9
c
50      WRITE(*,5)
5       FORMAT(///'  APPLY SUITABLY LARGE PRESSURE TO THE TRANSDUCER.'/
     *  ' THIS PRESSURE SHOULD BE GREATER THAN THE PRESSURE'/
     *  '      YOU ANTICIPATE DURING THE EXPERIMENT.'/////)
c
        IMODE = 0
        PARAM(1) = #300
        PARAM(2) = 2
        PARAM(3) = 3
        RCODE = DASH16(IMODE, PARAM)
c
        IMODE = 1
        PARAM(1) = 3
        PARAM(2) = 3
        RCODE = DASH16(IMODE, PARAM)
c
        MODE = 17
        PARAM(1) = 1000
        PARAM(2) = 20
        RCODE = DASH16(IMODE, PARAM)
c
        WRITE(*,51)
51      FORMAT(' **** PRESS TRIGGER TO COLLECT CALIBRATION ****'//)
c
        IMODE = 19
        PARAM(1) = 2
        PARAM(2) = 300
        PARAM(3) = 0
```

```
      RCODE = DASH16(IMODE, PARAM)
      WRITE(*,31)
      WRITE(*,4000) BELL
c
      IMODE = 4
      PARAM(1) = 12
      PARAM(2) = OFFADR(HIGH(0))
      PARAM(3) = 1
      RCODE = DASH16(IMODE, PARAM)
c
      C = 0.0
      WRITE(*,4)
      WRITE(*,4000) BELL
c
c Summing 10 peak pressures.
c
      DO 511 K=2,11
      HIG=HIGH(K)
      RHIGH(K)=FLOAT(HIG)
      C = C + RHIGH(K)
511   CONTINUE
c
c Calculating mean peak pressure.
c
      MEANHIGH = C / 10.0
      WRITE(*,52) (RHIGH(K),K=2,11),MEANHIGH
52    FORMAT(2(5(2X,F9.2)///),' MEAN HIGH PRESSURE = ',F9.2//)
      WRITE(*,53) MEANZERO
53    FORMAT(' MEAN ZERO PRESSURE = ',F9.2//)
c
      WRITE(*,54)
54    FORMAT(//' WHAT DO YOU WISH TO DO NEXT?'/
     *  ' 1 RETURN = LINEAR REGRESSION ANALYSIS,'/
     *  ' 6 RETURN = REPEAT POOR HIGH PRESSURE CALIBRATION,'/
     *  ' 9 RETURN = END Program.'//)
      READ(*,20) D

      IF (D.EQ.1) CONTINUE
      IF (D.EQ.6) GOTO 50
      IF (D.EQ.9) GOTO 9
c
      WRITE(*,6)
6     FORMAT(//' PLEASE TYPE THE PRESSURE OF THE SECOND CALIBRATION'/
     *  '      USING cm.H2O (e.g. xxxx.xx  [F7.2]) '//)
      READ(*,61) PRESSURE
61    FORMAT(F7.2)
c

c Linear regression analysis to obtain coefficients for
c equation: y = A + Bx.  Where:  A = y intercept,
c                               B = slope.
c
```

```
      W = 2 * PRESSURE * MEANHIGH
      X = PRESSURE * (MEANZERO + MEANHIGH)
      Y = 2 * PRESSURE * PRESSURE
      Z = PRESSURE * PRESSURE
      REGCOEFF = (W - X) / (Y - Z)
      V = MEANZERO + MEANHIGH
      CONSTANT = (V - REGCOEFF * PRESSURE) / 2
c
      WRITE(*,7) REGCOEFF,CONSTANT
7     FORMAT(////' LINEAR REGRESSION ANALYSIS HAS BEEN PERFORMED.'/
     *  '  REGRESSION COEFFICIENT B (cmH2O) = ',F9.2/
     *  '  CONSTANT A (cmH2O)               = ',F9.2//)
c
71    WRITE(*,72)
72    FORMAT(///'  TO CALCULATE ACTUAL PRESSURES USE THE EQUATION:'/
     *  '          x = (computer raw unit - A) / B.'///
     *  '          ENTER NAME OF PRESSURE CAL FILE:'//)
      WRITE(*,4000) BELL
      READ(*,73) FILENAME
73    FORMAT(A40)
c
      OPEN(3,FILE=FILENAME,STATUS='NEW')
      WRITE(3,8) REGCOEFF,CONSTANT
8     FORMAT(2X,F9.2,2X,F9.2)
      CLOSE(3)
c
821   WRITE(*,83)
83    FORMAT(///' WHAT DO YOU WISH TO DO NOW?'/
     *  '    1 RETURN = CALIBRATE THE MASS SPEC,'/
     *  '    6 RETURN = CALIBRATE PNEUMOTACHOGRAPH,'/
     *  '    9 RETURN = END Program.'//)
      READ(*,20) E
      IF (E.EQ.1) CONTINUE
      IF (E.EQ.6) GOTO 86
      IF (E.EQ.9) GOTO 9
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c               End-tidal CO2 calibration
c
c This calibration is exactly the same as that used for the mouth
c pressure calibration subroutines (computers are not smart enough
c to know the difference!!). Therefore no comments are provided.
c
84    IMODE = 0
      PARAM(1) = #300
      PARAM(2) = 2
      PARAM(3) = 3
      RCODE = DASH16(IMODE, PARAM)
c
      IMODE = 1
      PARAM(1) = 5
      PARAM(2) = 5
      RCODE = DASH16(IMODE, PARAM)
```

```
c
      IMODE = 17
      PARAM(1) = 1000
      PARAM(2) = 20
      RCODE = DASH16(IMODE, PARAM)
c
      WRITE(*,8400)
8400  FORMAT(///'       SET THE MASS SPEC OUTPUT TO ZERO.'/
     *  '       CHECK TRANSDUCER CONNECTED TO CHANNEL 5.'/
     *  '        ***** PRESS TRIGGER TO COMMENCE *****'//)
c
      IMODE=19
      PARAM(1)=2
      PARAM(2)=300
      PARAM(3)=0
      RCODE = DASH16(IMODE, PARAM)
c
      WRITE(*,31)
      WRITE(*,4000) BELL
c
      IMODE=4
      PARAM(1)=12
      PARAM(2)=OFFADR(BASE(0))
      PARAM(3)=1
      RCODE = DASH16(IMODE, PARAM)
c
      A = 0.00
      DO 841 J=2,11
      BAS=BASE(J)
      RBASE(J)=FLOAT(BAS)
      A = A + RBASE(J)
841   CONTINUE
c
      WRITE(*,4)
      WRITE(*,4000) BELL
      MEANZERO = A / 10.00
      WRITE(*,41) (RBASE(J),J=2,11),MEANZERO
      WRITE(*,42)
c
      WRITE(*,43)
      READ(*,20) G
      IF (G.EQ.1) CONTINUE
      IF (G.EQ.6) GOTO 84
      IF (G.EQ.9) GOTO 9
c
      WRITE(*,5111)
5111  FORMAT(//' APPLY SUITABLY HIGH CO2 OUTPUT FROM MASS SPEC.'//)
c
8420  IMODE = 0
      PARAM(1) = #300
      PARAM(2) = 2
      PARAM(3) = 3
```

```
      RCODE = DASH16(IMODE, PARAM)
c
      IMODE = 1
      PARAM(1) = 5
      PARAM(2) = 5
      RCODE = DASH16(IMODE, PARAM)
c
      IMODE = 17
      PARAM(1) = 1000
      PARAM(2) = 20
      RCODE = DASH16(IMODE, PARAM)
c
      WRITE(*,51)
c
      IMODE=19
      PARAM(1)=2
      PARAM(2)=300
      PARAM(3)=0
      RCODE = DASH16(IMODE, PARAM)
c
      WRITE(*,31)
      WRITE(*,4000) BELL
c
      IMODE=4
      PARAM(1)=12
      PARAM(2)=OFFADR(BIG(0))
      PARAM(3)=1
      RCODE = DASH16(IMODE, PARAM)
c
      WRITE(*,4)
      WRITE(*,4000) BELL
c
      C = 0.0
      DO 843 K=2,11
      BOG = BIG(K)
      RBIG(K)=FLOAT(BOG)
      C = C + RBIG(K)
843   CONTINUE
c
      MEANHIGH = C / 10.0
      WRITE(*,52) (RBIG(K),K=2,11),MEANHIGH
      WRITE(*,53) MEANZERO
c
      WRITE(*,54)
      READ(*,20) H
      IF (H.EQ.1) CONTINUE
      IF (H.EQ.6) GOTO 8420
      IF (H.EQ.9) GOTO 9
c
      WRITE(*,6111)
6111  FORMAT(///' ENTER VALUE OF HIGH % CO2:'//)
      READ(*,61) CONC2
```

```
c
      W = 2 * CONC2 * MEANHIGH
      X = CONC2 * (MEANZERO + MEANHIGH)
      Y = 2 * CONC2 * CONC2
      Z = CONC2 * CONC2
      REGCOEFF = (W - X) / (Y - Z)
      V = MEANZERO + MEANHIGH
      CONSTANT = (V - REGCOEFF * CONC2) / 2
c
      WRITE(*,7) REGCOEFF,CONSTANT
      WRITE(*,82)
82    FORMAT(///' ENTER NAME OF CALIBRATION FILE:'//)
      WRITE(*,4000) BELL
c
      READ(*,73) FILE2
      OPEN(3,FILE=FILE2,STATUS='NEW',FORM='FORMATTED')
      WRITE(3,8) REGCOEFF,CONSTANT
      CLOSE(3)
c
      WRITE(*,85)
85    FORMAT(///' WHAT DO YOU WISH TO DO NOW?'/
     *  '    1 RETURN = CALIBRATE PNEUMOTACHOGRAPH,'/
     *  '    6 RETURN = REPEAT BOTH LAST CALIBRATIONS,'/
     *  '    9 RETURN = END Program.'//)
      READ(*,20) F
      IF (F.EQ.6) GOTO 84
      IF (F.EQ.1) CONTINUE
      IF (F.EQ.9) GOTO 9
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c     Pneumotachograph calibration
c
c Initialising arrays LITER(J) & NOFLOW(J).
c
86    DO 8600 J=1,2002
      LITER(J)=0
8600  CONTINUE
      SAVE=0

c
      DO 8601 J=1,102
      NOFLOW(J)=0
8601  CONTINUE
c
      IMODE = 0
      PARAM(1) = #300
      PARAM(2) = 2
      PARAM(3) = 3
      RCODE = DASH16(IMODE, PARAM)
c
      IMODE =1
      PARAM(1) = 4
      PARAM(2) = 4
```

```
        RCODE = DASH16(IMODE, PARAM)
c
        IMODE = 17
        PARAM(1) = 1000
        PARAM(2) = 20
        RCODE = DASH16(IMODE, PARAM)
c
        WRITE(*,860)
860     FORMAT(//' CONNECT PNEUMOTACHOGRAPH TRANSDUCER TO CHANNEL FOUR.')
        WRITE(*,861)
861     FORMAT(///' *** PRESS TRIGGER TO COLLECT BASELINE DATA ***'//)
c
        IMODE=19
        PARAM(1)=2
        PARAM(2)=300
        PARAM(3)=0
        RCODE = DASH16(IMODE, PARAM)
c
        WRITE(*,31)
        WRITE(*,4000) BELL
c
        IMODE=4
        PARAM(1)=102
        PARAM(2)=OFFADR(NOFLOW(0))
        PARAM(3)=1
        RCODE = DASH16(IMODE, PARAM)
c
        WRITE(*,4000) BELL
4000    FORMAT('  ',A1)
        WRITE(*,4)
c
        A = 0
        DO 862 J=2,101
        A = A + NOFLOW(J)
862     CONTINUE
c
c  Writing to the screen all the basal flow data, plus their average.
c
        MEANNOFLOW = A / 100
        WRITE(*,863) (NOFLOW(J),J=2,101),MEANNOFLOW
863     FORMAT(10(10(1X,I5)/),' MEAN BASELINE FLOW = ',I5///)
c
        WRITE(*,42)
        WRITE(*,43)
        READ(*,20) G
        IF (G.EQ.1) CONTINUE
        IF (G.EQ.6) GOTO 86
        IF (G.EQ.9) GOTO 9
c
c  Saving the zero flow mean in a specific file.
c
        WRITE(*,82)
```

```
      WRITE(*,4000) BELL
      READ(*,73) FILE3
      OPEN(3,FILE=FILE3,STATUS='NEW',FORM='FORMATTED')
      WRITE(3,865) MEANNOFLOW
865   FORMAT(I7)
      CLOSE(3)
c
      WRITE(*,866)
866   FORMAT(///' CALIBRATE PNEUMOTACHOGRAPH USING 1 LITER SYRINGE.'/
     *  '              START FLOW AS SUCTION.'/
     *  '        USE SIX (6) PUMPS OF VARIABLE SPEED.'/
     *  ' DO NOT HIT THE PLUNGER OF THE SYRINGE HARD AGAINST'/
     *  ' THE ENDS OF ITS HOUSING, AS THIS PRODUCES POOR DATA.'///)
c
8651  IMODE = 0
      PARAM(1) = #300
      PARAM(2) = 2
      PARAM(3) = 3
      RCODE = DASH16(IMODE, PARAM)
c
      IMODE = 1
      PARAM(1) = 4
      PARAM(2) = 4
      RCODE = DASH16(IMODE, PARAM)
c
      IMODE = 17
      PARAM(1) = 1000
      PARAM(2) = 20
      RCODE = DASH16(IMODE, PARAM)
c
      WRITE(*,51)
c
      IMODE=19
      PARAM(1)=2
      PARAM(2)=300
      PARAM(3)=0
      RCODE = DASH16(IMODE, PARAM)
c
      WRITE(*,31)
      WRITE(*,4000) BELL
c
      IMODE=4
      PARAM(1)=2002
      PARAM(2)=OFFADR(LITER(0))
      PARAM(3)=1
      RCODE = DASH16(IMODE, PARAM)
c
      WRITE(*,4000) BELL
      WRITE(*,4)
      WRITE(*,867)
867   FORMAT(///' WHAT DO YOU WISH TO DO NOW?'/
     *  '      1 RETURN = SAVE DATA,'/
```

```
     *    '      6 RETURN = LOOK AT DATA.'/)
          READ(*,20) H
          IF (H.EQ.1) GOTO 870
          IF (H.EQ.6) GOTO 868
c
c  It is advisable to look at the data to check two things:
c  (1) that all syringe pumps are wholly present; and
c  (2) that no data points have flooded the A/D board. This occurs
c  when the flow signal produces an output greater than +/- 5 Volts.
c  The board only converts data in this range. All data outside this
c  range will be given maximal binary equivalents (i.e. 2048 or -2048).
c
868       WRITE(*,8680) (LITER(J),J=1,2000)
8680      FORMAT(10(I7))
c
864       WRITE(*,869)
869       FORMAT(///' WHAT DO YOU WISH TO DO NOW?'/
     *    '      1 RETURN = SAVE DATA,'/
     *    '      6 RETURN = REPEAT 6 LITER FLOW CALIBRATION,'/
     *    '      9 RETURN = END Program.'//)
          READ(*,20) H
          IF (H.EQ.1) CONTINUE
          IF (H.EQ.6) GOTO 8651
          IF (H.EQ.9) GOTO 9
c
870       WRITE(*,82)
          WRITE(*,4000) BELL
      READ(*,73) FILE4
c

c  Saving flow calibration data to a specific file.
c  These data are integrated to derive the calibration value
c  for the number of binary units equivalent to 1 litre. An example
c  of this is given following this program. We typically do this in
c  the analysis program, but it can be inserted here and saved instead
c  of the full flow calibration data. We save the full data in case we
c  get some unexpected errors which were not apparent during calibration.
c
          OPEN(3,FILE=FILE4,STATUS='NEW',FORM='UNFORMATTED')
          WRITE(3) (LITER(J),J=1,2000)
          SAVE=1
          CLOSE(3)
          WRITE(*,900)
900       FORMAT(///' CALIBRATION COMPLETED, AND ALL DATA SAVED'/
     *    '      WHAT DO YOU WISH TO DO NOW?'/
     *    '    1 RETURN = GO TO START OF Program'/
     *    '    9 RETURN = TERMINATE Program.'//)
          READ(*,20) ANS
          IF (ANS.EQ.1) GOTO 100
          IF (ANS.EQ.9) CONTINUE
c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
```

```
c  Subroutine that prevents program termination without first
c  saving the flow calibration data to a specific file.
c
9       IF (SAVE.LT.1) GOTO 864
        WRITE(*,9989)
9989    FORMAT(////////
     *  '  **********************************************'/
     *  '  *********                            *********'/
     *  '  *********     G O O D    B Y E      *********'/
     *  '  *********                            *********'/
     *  '  **********************************************'////)
        STOP
        END
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
```

PART II: Integration of the calibration flow signal.

Integration subroutine used with the flow calibration data to obtain the number of computer/binary units that are equivalent to 1 litre of the volume calibration standard. This subroutine may be seen in the composite program in Appendix 7.

```
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c  Pre-requisites for this subroutine include:
c   (1) calibration data filenames (NAME1 and NAME2);
c   (2) a set flow limit - this is the binary equivalent of typical
c          flow encountered during the experiment (this is variable
c          between experiments, and between amplifier and their gain
c          settings);
c   (3) a arbitrary volume limit - swallowing, sighing etc will produce
c          spurious volume excursions - this limit is used to prevent
c          these volumes being used during data analysis, but is also
c          used here to prevent artifacts from being included in the
c          calibration;
c  (4) a volume for the calibration syringe, determined by repeated
c          water filling.
c
        FILEBASE=NAME1
        FILEFLOW=NAME2
        FLOW=-500
        ARBITLIM=500
        SYRINGE = 3.000
c
c  Reading the zero flow calibration data from data disk.
c
        OPEN(3,FILE=FILEBASE,STATUS='OLD')
        READ(3,101) ZEROFLOW
        WRITE(*,102) ZEROFLOW
101     FORMAT(I7)
102     FORMAT(' BASAL FLOW (RAW UNITS) = ',I5//)
        CLOSE(3)
50      FORMAT(2X,F9.2,2X,F9.2)
c
c   Calculation of baselimit used to check for flow later
c   in program. This is pre-programd to suit anticipated
c   flow when volume is sucked slowly through the pneumotach to
c   give a negative signal. Transducer = Validyne ± 2.0 cmH2O.
c   The 4/100 value is used to set a 4% error level.
c   Two limits are set: (1) BASELIM for inspiration and (2) TBASELIM
c   for expiration. These serve as a zone above and below the actual
c   zero flow level within which the flow signal can oscillate before
c   triggering integration. This is very important in the analysis
c   subroutines since it minimizes the effect of artifacts without
c   compromising the integrity of the analysis.
```

```
c
      BASE = (FLOW - ZEROFLOW) * 4 / 100
      BASELIM = BASE + ZEROFLOW
      TBASELIM = ZEROFLOW - BASE
C     WRITE(*,1345) BASE
1345  FORMAT(' VALUE OF BASE = 'I5//)
      WRITE(*,104) BASELIM
      WRITE(*,104) TBASELIM
104   FORMAT(' VALUE OF BASELIMIT = 'I5//)
c
c Reading the flow calibration data.
c
      OPEN(3,FILE=FILEFLOW,STATUS='OLD',FORM='UNFORMATTED')
      READ(3) (LITER(J),J=1,2000)
11    FORMAT(10(I7))
      CLOSE(3)
c
c   Integration of flow to derive volume.
c   L will represent the breath counter (i.e. # of breaths).
c   J will represent the sample number currently analysed.
c
c   Checking whether flow is baseline or a calibration input.
c
      M=1
      L=0
12    DO 120 J=M,2000
      IF (LITER(J).LT.BASELIM) GOTO 1200
120   CONTINUE
1200  CONTINUE
c
      IF (J.GE.2000) GOTO 129
c
c  Integration of calibration flow data.
c
      VOLINSP=0.0
c
121   DO 127 T=J,2000
      IVOL=LITER(T)-ZEROFLOW
      RVOL=FLOAT(IVOL)
      VOLINSP=VOLINSP-RVOL
      IF (LITER(T).GE.BASELIM) GOTO 1270
127   CONTINUE
1270  CONTINUE
c
c   Checking data for false volumes before storing volumes.
c
      IF (VOLINSP.LE.ARBITLIM) GOTO 128
      L=L+1
      CALVOL(L)=VOLINSP
128  IF (T.GE.2000) GOTO 129
      M=T
      GOTO 12
```

```
c
c  Calculation of average calibration value (binary units per litre).
c
129     RVOLSUM=0.0
        DO 1280 W=1,L
        RVOLSUM=RVOLSUM+CALVOL(W)
1280    CONTINUE
c
        VOLAVE = RVOLSUM / (SYRINGE*L)
1281    FORMAT(10(I7))
        WRITE(*,1282) VOLAVE
1282    FORMAT(' MEAN FLOW CALIBRATION VALUE =',F14.4,' units/litre'//)
c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
```

## APPENDIX 3

Respiratory pattern subroutines.

Respiratory pattern subroutines: These routines perform two functions: (1) compute tidal volumes from data obtained during rest or exercise; (2) compute all respiratory timing (including pauses) data and derive minute ventilation. These routines are contained within the composite example in Appendix 7.

```
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c  Initializing variables
c
        DO 3002 J=1,40
        TIDALV(J)=0.0
        VOLBTPS(J)=0.0
        IONSET(J)=0
        EONSET(J)=0
        TINSP(J)=0.0
        TEXP(J)=0.0
        TTOTAL(J)=0.0
3002    CONTINUE
c
        TOTALVOL = 0.0
c
c  COUNTERS: I = inspiration onset, E = expiration onset, [onset
c                occurs when the flow signal crosses a limit threshold]
c          P = pairs of insp. and exp., M = temporary marker,
c          L = tidal volumes.
c  NOTE: I is used as a respiratory cycle counter from this point.
c  Integrating flow to obtain volume (inspiration -ive).
c
        I=0
        E=0
        M=1
        L=0
c
c  Here we are checking to make sure we start the analysis with
c  a signal in the inspiratory phase [convention: inspiration
c  produces a negative flow signal]. The program doesn't move forward
c  till the desired point is achieved in the flow signal.
c
3       DO 3111 J=M,MAX
        IF (PNEUMO(J).GE.BASELIM) GOTO 3112
3111    CONTINUE
3112    DO 31 K=J,MAX
        IF (PNEUMO(K).LT.BASELIM) GOTO 310
31      CONTINUE
c
310     IF (K.GE.MAX) GOTO 3102
c
```

```
C       WRITE(*,5500) K,PNEUMO(K)
C5500   FORMAT(' IONSET AT POINT ',I5,' = ',I5)
C
        VOLINSP=0.0
        VOL2INSP=0.0
        EXVOLINSP=0.0
        EX2VOLINSP=0.0
c
c  Since we know the signal is negative (above), we can now look for
c  the next onset of inspiration. This ensures that the first inspiration
c  is trully a complete breath and not partially cut by failure of the
c  subject to coincide inspiration onset with the start of data collection.
c  Integration of inspiratory flow (addition, ie VOLINSP-(-RVOL)).
c
        DO 32 T=K,MAX
        IF (PNEUMO(T).GE.BASELIM) GOTO 33
        IVOL=PNEUMO(T)-BASEFLOW
        RVOL=FLOAT(IVOL)
        VOLINSP = VOLINSP - RVOL
32      CONTINUE
33      CONTINUE
c
c  The -RVOL converts volume to positive.
c  Check for small, false volumes.
c
c       WRITE(*,5501) VOLINSP
C5501   FORMAT(' INTEGRATED TOTAL = ',F10.2)
c
        IF (VOLINSP.LT.ARBITLIM) GOTO 3103
c
c   Marking points of inspiration start (IONSET), & end (EONSET),
c   and expiration start (STARTEXP), & end (ENDEXP).
c
        I=I+1
        IONSET(I)=K
        EONSET(I)=T
        DO 330 J=1,15
        VOL(J)=0.0
330     CONTINUE
c
    L=L+1
c
c  Applying the calibration value (VOLAVE: see Appendix 2).
c
        VOL(L) = VOLINSP / VOLAVE


c
c    Volume correction for regulator pressure and PLC, then BTPS
c    correction (all values are in cmH2O and degC units).
c    Variables: PREG = regulator pressure at the center of the diaphragm
c    (if needed), PH2O = water vapour pressure at TA, TA = ambient temperature,
```

```
c     PLC = lung centroid pressure (if needed), 64.056 = 47.1*1.36 (cmH2O),
c     TA2 = temperature of expired gas (heater element involved), PH2O2 =
c     expired water vapour pressure at temp TA2.
c
          VOL(L) = VOL(L) * (PBAR+PREG) / (PBAR)
          VOLBTPS(L)=VOL(L)*310/(273+TA)*(PBAR+PLC-PH2O)
     *          /(PBAR+PLC-64.056)
          TIDALV(L)=VOLBTPS(L)
c
3103      IF (T.GE.MAX) GOTO 3102
          IF (VOLINSP.GT.ARBITLIM) GOTO 3104
          M = T
          GOTO 3
c
c This process is now duplicated for the expiratory phase. This phase
c may be omitted if only inspiratory data are collected.
c
3104      DO 34 EX=T,MAX
          IF (PNEUMO(EX).GT.TBASELIM) GOTO 340
34        CONTINUE
340       CONTINUE
c
          DO 341 EXP=EX,MAX
          IF (PNEUMO(EXP).LE.TBASELIM) GOTO 3410
          EVOL=PNEUMO(EXP)-BASEFLOW
          ERVOL=FLOAT(EVOL)
          EXVOLISNP = EXVOLINSP + ERVOL
341       CONTINUE
3410      CONTINUE
c
          IF (EXP.GE.MAX) GOTO 3102
          IF (EXVOLINSP.GT.ARBITLIM) GOTO 3411
          T = EXP
          GOTO 3104
C
3411      E=E+1
          STARTEXP(E)=EX
          ENDEXP(E)=EXP
c
          EXTIDALV(L)=EXVOLINSP/VOLAVE*(PBAR+PREG)/(PBAR)*310/(273+TA2)
     *          *(PBAR+PLC-PH2O2)/(PBAR+PLC-64.056)
c


          M=T
          GOTO 3
3102      CONTINUE
c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
c     RESPIRATORY TIMING SUBROUTINES (including post-insp
c       [POSTINSP] and post-expiratory pauses [POSTEXP]).
```

```
c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
c  Computing timing/pausing periods based upon a sampling rate of
c  50 Hz. The error is this calculation is 20 milliseconds, and can
c  be reduced by increasing the sampling rate.
c  Variables: IONSET = start of inspiration, EONSET = end of inspiration,
c  STARTEXP = start of expiration, ENDEXP = end of expiration, POSTINSP =
c  post-inspiratory pause, POSTEXP = post-expiratory pause
c
        DO 42 P=1,(I-1)
        TINSP(P) = (EONSET(P) - IONSET(P)) / 50.0
        TEXP(P) = (ENDEXP(P)-STARTEXP(P))/50.0
        TTOTAL(P) = (IONSET(P+1)-IONSET(P))/50.0
        POSTINSP(P) = (STARTEXP(P)-EONSET(P))/50.0
        POSTEXP(P) = (IONSET(P+1)-ENDEXP(P))/50.0
42      CONTINUE
c
        WRITE(*,103) I,E
103     FORMAT('    ',I5,' INSPIRATIONS AND ',I3,' EXPIRATIONS'///)
        WRITE(*,420)
420     FORMAT(' TIDAL/I TIDAL/E INSP/Time EXP/Time T/tot '
     *  ' POST-INSP/Pause POST-EXP/Pause')
c
        DO 421 P=1,(I-1)
        WRITE(*,422) TIDALV(P),EXTIDALV(P),TINSP(P),TEXP(P),TTOTAL(P),
     *  POSTINSP(P),POSTEXP(P)
421     CONTINUE
422     FORMAT(F7.3,2X,F7.3,2X,F7.4,1X,F7.4,1X,F7.4,5X,F7.4,8X,F7.4)
c
        AVTINSP=0.0
        AVTEXP=0.0
        AVTTOTAL=0.0
        AVTIDALV = 0.0
        AVINSPPOST = 0.0
        AVEXPPOST = 0.0
c
c  Summing the timing components
c
                    .


        DO 44 P=1,(I-1)
        AVTIDALV = AVTIDALV + TIDALV(P)
        AVTINSP = AVTINSP + TINSP(P)
        AVTEXP = AVTEXP + TEXP(P)
        AVTTOTAL = AVTTOTAL + TTOTAL(P)
        AVINSPPOST = AVINSPPOST + POSTINSP(P)
        AVEXPPOST = AVEXPPOST + POSTEXP(P)
44      CONTINUE
c
c  Computing averages for each timing component.
```

```
c
      AVTIDALV = AVTIDALV / (I-1)
      AVTINSP = AVTINSP / (I-1)
      AVTEXP = AVTEXP / (I-1)
      AVTTOTAL = AVTTOTAL / (I-1)
      RFREQ=60.0/AVTTOTAL
      VTTIMEINSP = AVTIDALV / AVTINSP
      MINVENT = AVTIDALV * RFREQ
      TINTOTAL = AVTINSP / AVTTOTAL
      AVINSPPOST = AVINSPPOST / (I-1)
      AVEXPPOST = AVEXPPOST / (I-1)
c
      WRITE(*,440) AVTIDALV
440   FORMAT(//////' AVERAGE TIDAL VOLUME = ',F12.4/)
      WRITE(*,441) AVTINSP
441   FORMAT(' AVERAGE TIME INSP = ',F12.4)
      WRITE(*,442) AVTEXP
442   FORMAT(' AVERAGE TIME EXPIR. = ',F12.4)
      WRITE(*,443) AVTTOTAL
443   FORMAT(' AVERAGE TOTAL TIME = ',F12.4/)
      WRITE(*,444) VTTIMEINSP
444   FORMAT('    VT/Ti =    ',F12.4)
      WRITE(*,446) TINTOTAL
      WRITE(*,4400) RFREQ
4400  FORMAT(' BREATHING FREQUENCY = ',F12.4)
      WRITE(*,445) MINVENT
445   FORMAT(' MINUTE VENT. = ',F12.4/)
446   FORMAT(' Ti/Ttot = ',F12.4/)
      WRITE(*,447) AVINSPPOST
447   FORMAT(' AVERAGE POST-INSPIRATORY PAUSE = ',F12.4)
      WRITE(*,448) AVEXPPOST
448   FORMAT(' AVERAGE POST-EXPIRATORY PAUSE = ',F12.4)
c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
```

## APPENDIX 4

Lung volume subroutines.

The determination of the various lung volumes is first dependent upon the analyses performed in Appendix 3, since each of the volumes and capacities shall be considered as a separate tidal volume (TIDALV). This subroutine is contained within the example program in Appendix 7.

```
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
      WRITE(*,3337)
      DO 3339 P=1,I
      WRITE(*,3338) P, TIDALV(P)
3339  CONTINUE
3337  FORMAT('  VOLUME #   SPIROMETRIC VOLUMES (inspiratory)')
3338  FORMAT(2X,I4,11X,F7.3)
c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
```

## APPENDIX 5

Maximal voluntary ventilation subroutine.

Calculations performed here are also dependent upon those performed in Appendix 3, since each tidal volume (TIDALV) is added for the duration of the maneuver, and the duration from the start to the end of the maneuver are required (MVVTIME). This subroutine is contained within the example program in Appendix 7.

```
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
      MVV = 0.0
c
      DO 35 P=1,(I-1)
      MVV = MVV + TIDALV(P)
35    CONTINUE
c
c  The number 50.0 and 60.0 are used because the sampling rate was 50 Hz and the
c  output is required in liters per minute (60.0).
c
      MVVTIME = (ENDEXP(I-1)-IONSET(1)) / 50.0 / 60.0
      MVV = MVV / MVVTIME
      WRITE(*,350) MVV
350   FORMAT(///'   MAXIMAL VOLUNTARY VENTILATION FOR THIS'/
     *  '       TRIAL = ',F12.4,' Litres BTPS'///)
c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
```

## APPENDIX 6

### Maximal flow-volume loop subroutines.

Calculations performed here are also dependent upon those performed in Appendix 3, since the onset of each of the phases of respirations are used as markers for the analysis. This subroutine is contained within the example program in Appendix 7. Several variables have previously been explained in other Appendices and their explanations are not repeated.

```
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
c  Convention for flow signs during analysis: inspiration -ive,
c  expiration +ive.
c
c  In these routines all respiratory cycles are analysed, thus
c  in DO 6 P=1,I and not I-1. Requires a data output filename (FILE2).
c  TOPVOL = the final lung volume or TLC, RVOLIN = integrated inspired
c  volume, VOLEX = integrated expired volume, RINFLOW = inspiratory flow
c  variable, REXFLOW = expiratory flow variable.
c
        OPEN(4,FILE=FILE2,STATUS='NEW')
        TOPVOL=0.0
c
        DO 6 P=1,I
c
c   Setting arrays and variables to zero.
c
            DO 60 J=1,200
               RVOLIN(J)=0.0
               VOLEX(J)=0.0
               RINFLOW(J)=0.0
               RVOLEX(J)=0.0
               REXFLOW(J)=0.0
60          CONTINUE
c
c  Computes the inspiratory volume and flow at each 20 millisecond
c  interval. These data are written to both the screen and output file.
c
                WRITE(*,63)
            DO 610 PPP=IONSET(P),(EONSET(P)-1)
               PP=(PPP-IONSET(P))+1
               INVOL=PNEUMO(PPP)-BASEFLOW
               RINVOL=FLOAT(INVOL)
               RINVOL=RINVOL/VOLAVE*(PBAR+PREG)/(PBAR)*310/
     *         (273+TA)*(PBAR+PLC-PH2O)/(PBAR+PLC-64.056)
               IF (RINVOL.LT.(0.0)) GOTO 6100
               RINVOL=0.0-RINVOL
```

```
6100          RVOLIN(PP)=RVOLIN(PP-1)-RINVOL
              RINFLOW(PP)=FLOAT(INVOL)
              RINFLOW(PP)=RINFLOW(PP)/VOLAVE*50.0*(PBAR+PREG)/(PBAR)
     *       *310/(273+TA)*(PBAR+PLC-PH2O)/(PBAR+PLC-64.056)
              WRITE(*,631) PPP,RVOLIN(PP),RINFLOW(PP)
              WRITE(4,631) PPP,RVOLIN(PP),RINFLOW(PP)
610         CONTINUE
              TOPVOL=RVOLIN(PP)
              WRITE(*,630)
c
c Expiratory data analysis.
c
            DO 620 PPP=STARTEXP(P),(ENDEXP(P)-1)
              PP=(PPP-STARTEXP(P))+1
              EVOL=PNEUMO(PPP)-BASEFLOW
              REXVOL=FLOAT(EXVOL)
              REXVOL=REXVOL/VOLAVE*(PBAR+PREG)/(PBAR)*310/
     *          (273+TA2)*(PBAR+PLC-PH2O2)/(PBAR+PLC-64.056)
              RVOLEX(PP)=RVOLEX(PP-1)+REXVOL
              VOLEX(PP)=TOPVOL-RVOLEX(PP)
              REXFLOW(PP)=FLOAT(EXVOL)
              REXFLOW(PP)=REXFLOW(PP)/VOLAVE*50.0*(PBAR+PREG)/(PBAR)
     *          *310/(273+TA2)*(PBAR+PLC-PH2O2)/(PBAR+PLC-64.056)
              WRITE(*,631) PPP,VOLEX(PP),REXFLOW(PP)
              WRITE(4,631) PPP,VOLEX(PP),REXFLOW(PP)
620         CONTINUE
6           CONTINUE
c
63          FORMAT(' INSPIRATORY DATA'/
     *      ' #  Vol(BTPS)  Flow  ')
630         FORMAT(/' EXPIRATORY DATA'/
     *      ' #  Vol(BTPS)  Flow  ')
631         FORMAT(150(I4,1X,F8.4,2X,F9.4))
            CLOSE(4)
c
c Subroutines for computing FEV1.0 from the data obtained
c during the maximal flow-volume loop trials.
c
            EVOL=0
            FEV1=0.0
            EXVOLINSP=0.0
            DO 64 P=STARTEXP(1),(STARTEXP(1)+49)
                  EVOL=PNEUMO(P)-BASEFLOW
                  ERVOL=FLOAT(EVOL)
                  EXVOLINSP=EXVOLINSP+ERVOL
64          CONTINUE
c
```

```
      FEV1=EXVOLINSP/VOLAVE*(PBAR+PREG)/(PBAR)*310
     *  /(273+TA2)*(PBAR+PLC-PH2O2)/(PBAR+PLC-64.056)
      WRITE(*,65) FEV1
65    FORMAT(///' FEV1 = ',F8.4,' Litres BTPS.'///)
c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
```

## APPENDIX 7

Example of a composite data analysis program.

Many of the comments necessary to understand the following program and its variables have been omitted, since they have been incorporated in the Appendices above.

```
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
c  PROGRAM TO ANALYSE DATA FROM RESPIRATORY VOLUME EXPERIMENTS.
c
c Program uses data obtained from A/D program SPIRO-D.EXE
c using the DASH16 A/D board (MetraByte); (50 Hz, 2 channels)
c
c          INPUT DATA REQUIRED: Ch2=Trigger, Ch3=flow
c       Flow signal is both inspiratory and expiratory
c
c  NOTE: THIS Program HAS OUTPUT IN S.I. UNITS
c       THE ANALYSIS RELIES UPON USING A HEATED PNEUMO @ 37oC
c
c            THE FOLLOWING ANALYSES ARE PERFORMED:
c     (1) Tidal volumes, (2) Respiratory timing (Ti, Te, Ttot,
c     Vt/Ti, Ti/Ttot, fb, Ve, and pauses), (3) Maximal Voluntary
c     Ventilation, (4) Flow-Volume loops and FEV1, (5) Spirometry
c     for determining lung volumes and capacities.
c
c          NIGEL A.S. TAYLOR.          MAY 1989
c          Diving Life Support Equipment Program
c              Naval Medical Research Institute
c          Bethesda, MD.
c
c  LATEST UPDATE: 4/23/90
c
c  This program allows for unequal inspired/expired volumes
c     by using only complete cycles, starting with an inspiration.
c     Inspiration is detected from a negative signal from the flow
c     channel.
c
c  Program is able to analyse data obtained when subjects are immersed
c     and air is provided at various hydrostatic pressures. This requires
c     modification after completion of the calibration analysis routines.
c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
$STORAGE:2
c
        INTEGER ZEROFLOW,LITER,BASELIM,PHASE1,PHASE2,INVOL
        INTEGER PNEUMO,NOFLOW,FLOW,BASE,IDATA,COMP,EXVOL
        INTEGER GAIN,PREG,PSTERN,IONSET,EONSET,TIME,MAX
        INTEGER MAX2,IVOL,BASEFLOW,PRESS,INFLOW,EXFLOW
        INTEGER A,B,C,D,E,F,I,J,K,L,M,N,P,PP,PPP,Q,R,S,T,W,X,Y,Z
```

```
      INTEGER TBASELIM,EX,EXP,ENDEXP,STARTEXP,EVOL,REPEAT
c
      REAL VOLUME,VOLAVE,RVOLSUM,PH2O,TA,AVTIDALV,TINSP
      REAL TOTALVOL,INSPVOL,SYRINGE,VOL,TIDALV,TEXP,TTOTAL
      REAL VOLINSP,CALVOL,RVOL,ARBITLIM,VTTIMEINSP,AVTINSP
      REAL AVTEXP,AVTTOTAL,RFREQ,MINVENT,MINV2,TINTOTAL,TIDE
      REAL ERVOL,MVV,MVVTIME,POSTINSP,POSTEXP,TOPVOL,TA2
      REAL COMPRESS,PLC,SIPLC,REXVOL,VOLEX,RVOLEX,PH2O2
      REAL RINVOL,RVOLIN,RINFLOW,REXFLOW,VOL2INSP,EX2VOLINSP
      REAL INT,EXT,EXTIDALV,EXSMALLTV,EXTIDE
      REAL EXVOLINSP,AVINSPPOST,AVEXPPOST,FEV1
      REAL*4 PBAR
c
      DIMENSION IDATA(2,6000),LITER(2000)
      DIMENSION PNEUMO(3000),VOL(40),VOLBTPS(40),IONSET(40),TINSP(40)
      DIMENSION CALVOL(10),EONSET(40),TIDALV(40),TEXP(40),TTOTAL(40)
      DIMENSION POSTINSP(40),POSTEXP(40),PINSP(40)
      DIMENSION RVOLIN(200),RINFLOW(200),RVOLEX(200),VOLEX(200)
      DIMENSION REXFLOW(200),TIDE(40)
      DIMENSION EXTIDALV(40),EXSMALLTV(40),EXTIDE(40)
      DIMENSION ENDEXP(40),STARTEXP(40)
c
      CHARACTER BELL
      CHARACTER*40 FILEBASE,FILEFLOW,FILE1,FILE2
      CHARACTER*40 NAME1,NAME2,NAME3,NAME4
      BELL = CHAR(7)
      OPEN(*)
c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
c     INTEGRATION OF FLOW TO GET TIDAL VOLUMES.
c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
      REPEAT = 0
      TIME = 0
      WRITE(*,7000)
7000  FORMAT(/////////
     *  '       RESPIRATORY VOLUME ANALYSIS Program:'////
     *  '   DO YOU WISH TO USE DEFAULT CALIBRATION FILES?'/
     *  '            1 RETURN = YES,'/
     *  '            9 RETURN = NO.'/)
      READ(*,100) A
100   FORMAT(I3)
      IF (A.NE.1) GOTO 7001
      NAME1='VOLBASE.DAT'
      NAME2='VOLCALIB.DAT'
      NAME3='VOL2BASE.DAT'
      NAME4='VOL2CAL.DAT'
      GOTO 7002
7001  WRITE(*,7003)
7003  FORMAT(//'  ENTER FOUR (4) CALIBRATION FILENAMES.'///)
```

```
      WRITE(*,7004)
7004  FORMAT(' ENTER LOW GAIN BASAL CALIBRATION FILENAME:'/)
      READ(*,7005) NAME1
7005  FORMAT(A40)
      WRITE(*,7006)
7006  FORMAT(' ENTER LOW GAIN FLOW CALIBRATION FILENAME:'/)
      READ(*,7005) NAME2
      WRITE(*,7007)
7007  FORMAT(' ENTER HIGH GAIN BASAL CALIBRATION FILEMANE:'/)
      READ(*,7005) NAME3
      WRITE(*,7008)
7008  FORMAT(' ENTER HIGH GAIN FLOW CALIBRATION FILENAME:'/)
      READ(*,7005) NAME4
7002  CONTINUE
1310  WRITE(*,1)
1     FORMAT(///
     *  '     WHICH TYPE OF DATA DO YOU WISH TO ANALYSE?'/
     *  '         1 RETURN = SPONTANEOUS RESPIRATION'/
     *  '         3 RETURN = LUNG VOLUME SPIROMETRY'/
     *  '         6 RETURN = MAXIMAL VOLUNTARY VENTILATION'/
     *  '         9 RETURN = MAXIMAL FLOW-VOLUME LOOPS'/)
      READ(*,100) A
C
c  Determining data file dimensions and setting basal input
c   according to the gain settings used for different experiments
C
      IF (A.NE.1.AND.A.NE.3) GOTO 1991
      IF (A.EQ.1) TIME=60*50*2
      IF (A.EQ.3) TIME=45*50*2
      FILEBASE=NAME1
      FILEFLOW=NAME2
      FLOW=-300
      ARBITLIM=1500
      GOTO 1990
1991  IF (A.EQ.6) TIME=15*50*2
      IF (A.EQ.9) TIME=12*50*2
      FILEBASE=NAME3
      FILEFLOW=NAME4
      FLOW=-500
      ARBITLIM=500
1990  MAX2=TIME
      MAX=MAX2/2
C
      SYRINGE = 3.000
      OPEN(3,FILE=FILEBASE,STATUS='OLD')
      READ(3,101) ZEROFLOW
      WRITE(*,102) ZEROFLOW
101   FORMAT(I7)
102   FORMAT(' BASAL FLOW (RAW UNITS) = ',I5//)
      CLOSE(3)
50    FORMAT(2X,F9.2,2X,F9.2)
```

```
      BASE = (FLOW - ZEROFLOW) * 4 / 100
      BASELIM = BASE + ZEROFLOW
      TBASELIM = ZEROFLOW - BASE
      WRITE(*,1345) BASE
1345  FORMAT(' VALUE OF BASE = '15//)
      WRITE(*,104) BASELIM
      WRITE(*,104) TBASELIM
104   FORMAT(' VALUE OF BASELIMIT = '15//)
c
      OPEN(3,FILE=FILEFLOW,STATUS='OLD',FORM='UNFORMATTED')
      READ(3) (LITER(J),J=1,2000)
11    FORMAT(10(I7))
      CLOSE(3)
c
      M=1
      L=0
12    DO 120 J=M,2000
      IF (LITER(J).LT.BASELIM) GOTO 1200
120   CONTINUE
1200  CONTINUE
c
      IF (J.GE.2000) GOTO 129
      VOLINSP=0.0
c
121   DO 127 T=J,2000
      IVOL=LITER(T)-ZEROFLOW
      RVOL=FLOAT(IVOL)
      VOLINSP=VOLINSP-RVOL
      IF (LITER(T).GE.BASELIM) GOTO 1270
127   CONTINUE
1270  CONTINUE
c
      IF (VOLINSP.LE.ARBITLIM) GOTO 128
      L=L+1
      CALVOL(L)=VOLINSP
128   IF (T.GE.2000) GOTO 129
      M=T
      GOTO 12
c
129   RVOLSUM=0.0
      DO 1280 W=1,L
      RVOLSUM=RVOLSUM+CALVOL(W)
1280  CONTINUE
c
      VOLAVE = RVOLSUM / (SYRINGE*L)
1281  FORMAT(10(I7))
      WRITE(*,1282) VOLAVE
1282  FORMAT(' MEAN FLOW CALIBRATION VALUE ='F14.4,' units/litre'//)
c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
      IF (REPEAT.EQ.1) GOTO 1311
```

```
C
2       FORMAT(//'  PLEASE NAME THE DATA FILE TO BE ANALYSED.'///
     *  '    examples: (1) BARNES.RES = SPONTANEOUS BREATHING'/
     *  '              (3) BARNES.SPI = LUNG VOLUME SPIROMETRY'/
     *  '              (6) BARNES.MVV = MAXIMAL VOLUNTARY VENTILATION'/
     *  '              (9) BARNES.FLO = MAXIMAL FLOW-VOLUME LOOPS'///)
20      FORMAT(A40)
c
        WRITE(*,130)
130     FORMAT(' WHAT WAS THE BAROMETRIC PRESSURE (mmHg F8.4)?'/)
        READ(*,131) PBAR
131     FORMAT(F12.3)
        PBAR = PBAR * 1.36
c
c The question below is used only if inspired air temp was not 37 °C
c  TA = inspired gas temperature, TA2 = expired temp, PH2O = water
c  vapour pressure, PH2O2 = expired water vapour pressure. Both pressures
c  are converted to cmH2O.
c
C       WRITE(*,132)
C132    FORMAT(' WHAT WAS THE INSPIRED AIR TEMPERATURE (deg F F8.4)?'/)
C       READ(*,131) TA
        TA = 37.0
        TA2 = 37.0
        PH2O2 = 47.1 * 1.36
        WRITE(*,133)
133     FORMAT(' WHAT WAS INSPIRED WATER VAPOR PRESSURE (mmHg F8.4)?'/)
        READ(*,131) PH2O
        PH2O = PH2O * 1.36
c
c Variables used when air is provided at various breathing pressures.
c COMP is compensator pressure used to balance the differential transducer.
c This is necessary when mouth pressures during immersion exceed the range
c limits of the transducer. PLC = lung centroid pressure, COMPRESS and
c SIPLC are SI equivalents of the above pressures. PREG = hydrostatic
c pressure at center of the regulator (ie. depth in cmH2O), PSTERN =
c hydrostatic pressure at the depth of the sternal notch, PHASE = the
c phase shift between flow and pressure signals (not used).
c

        COMP = 0
        COMPRESS= 0.0
        PLC = 0
        SIPLC = 0
        PREG = 0
        PSTERN = 0
        PHASE = 0
c
1311    WRITE(*,2)
        READ(*,20) FILE1
c
c  The output file is in ASCII format, and with the extension .PRN
```

```
c  it may be IMPORTED directly into a LOTUS123 worksheet. This is
c  particularly useful when analysing the flow-volume loops.
c
       WRITE(*,1340)
1340   FORMAT('  WHAT DO YOU WISH TO CALL THE OUTPUT FILE?'//
     *  '         example:  BARN-RES.PRN  '///)
       READ(*,20) FILE2
c
135    FORMAT(I3)
c
c  Reading unformatted data file
c
       OPEN(3,FILE=FILE1,STATUS='OLD',FORM='UNFORMATTED')
       DO 21 K=1,2
       READ(3) (IDATA(K,L),L=1,MAX2)
21     CONTINUE
       CLOSE(3)
c
c  Placing flow data into the array PNEUMO.
c
       DO 22 J=1,MAX
       PNEUMO(J)=IDATA(2,J)
22     CONTINUE
24     FORMAT(////'    DATA FILE ANALYSED = ',A40////)
c
c    Volume analysis routine.
c    Check baseline again, and use same baselimit value.
c    These next statements are used to check exp. data for basal
c    points.
c
       N=0
       DO 30 J=5,14
       N=N+PNEUMO(J)
30     CONTINUE
       NOFLOW = N / 10
C      WRITE(*,301) NOFLOW
C301   FORMAT(' BASAL FLOW FROM TRIAL (raw units) = ',I5/)
c
       WRITE(*,355) BELL
355    FORMAT(' ',A1)
       WRITE(*,3000)
3000   FORMAT(///'  Activate printer if print-out required'///
     *  '           1 RETURN = TO CONTINUE'/)
       READ(*,100) F
       WRITE(*,24) FILE1
c
       BASEFLOW=ZEROFLOW
c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
       DO 3002 J=1,40
       TIDALV(J)=0.0
```

```
        VOLBTPS(J)=0.0
        IONSET(J)=0
        EONSET(J)=0
        TINSP(J)=0.0
        TEXP(J)=0.0
        TTOTAL(J)=0.0
3002    CONTINUE
c
        TOTALVOL = 0.0
c
        I=0
        E=0
        M=1
        L=0
c
3       DO 3111 J=M,MAX
        IF (PNEUMO(J).GE.BASELIM) GOTO 3112
3111    CONTINUE
3112    DO 31 K=J,MAX
        IF (PNEUMO(K).LT.BASELIM) GOTO 310
31      CONTINUE
c
310     IF (K.GE.MAX) GOTO 3102
c
C       WRITE(*,5500) K,PNEUMO(K)
C5500   FORMAT(' IONSET AT POINT ',I5,' = ',I5)
C
        VOLINSP=0.0
          VOL2INSP=0.0
        EXVOLINSP=0.0
          EX2VOLINSP=0.0
c


        DO 32 T=K,MAX
        IF (PNEUMO(T).GE.BASELIM) GOTO 33
        IVOL=PNEUMO(T)-BASEFLOW
        RVOL=FLOAT(IVOL)
        VOLINSP = VOLINSP - RVOL
32      CONTINUE
33      CONTINUE
c
C       WRITE(*,5501) VOLINSP
C5501   FORMAT('  INTEGRATED TOTAL = ',F10.2)
c
        IF (VOLINSP.LT.ARBITLIM) GOTO 3103
c
        I=I+1
        IONSET(I)=K
          EONSET(I)=T
        DO 330 J=1,15
        VOL(J)=0.0
```

```
330     CONTINUE
c
        L=L+1
        VOL(L) = VOLINSP / VOLAVE
c
c   Volume correction for regulator pressure and PLC, then BTPS correction
c   (all values are in cmH2O and degC units).
c
        VOL(L) = VOL(L) * (PBAR+PREG) / (PBAR)
        VOLBTPS(L)=VOL(L)*310/(273+TA)*(PBAR+PLC-PH2O)
     *      /(PBAR+PLC-64.056)
        TIDALV(L)=VOLBTPS(L)
c
3103    IF (T.GE.MAX) GOTO 3102
        IF (VOLINSP.GT.ARBITLIM) GOTO 3104
        M = T
        GOTO 3
c
3104    DO 34 EX=T,MAX
        IF (PNEUMO(EX).GT.TBASELIM) GOTO 340
34      CONTINUE
340     CONTINUE
c
        DO 341 EXP=EX,MAX
        IF (PNEUMO(EXP).LE.TBASELIM) GOTO 3410
        EVOL=PNEUMO(EXP)-BASEFLOW
        ERVOL=FLOAT(EVOL)
          EXVOLISNP = EXVOLINSP + ERVOL
341     CONTINUE
3410    CONTINUE
c

        IF (EXP.GE.MAX) GOTO 3102
        IF (EXVOLINSP.GT.ARBITLIM) GOTO 3411
        T = EXP
        GOTO 3104
c
3411    E=E+1
        STARTEXP(E)=EX
        ENDEXP(E)=EXP
c
    EXTIDALV(L)=EXVOLINSP/VOLAVE*(PBAR+PREG)/(PBAR)*310/(273+TA2)
   *        *(PBAR+PLC-PH2O2)/(PBAR+PLC-64.056)
c
    M=T
        GOTO 3
3102    CONTINUE
c
    IF (A.EQ.9) GOTO 6000
c
c       Maximal voluntary ventilation calculations
c
```

```
      MVV = 0.0
      IF (A.NE.6) GOTO 3333
      DO 35 P=1,(I-1)
      MVV = MVV + TIDALV(P)
35    CONTINUE
      MVVTIME = (ENDEXP(I-1)-IONSET(1)) / 50.0 / 60.0
      MVV = MVV / MVVTIME
      WRITE(*,350) MVV
350   FORMAT(///'   MAXIMAL VOLUNTARY VENTILATION FOR THIS'/
     *  '       TRIAL = ',F12.4,' Litres BTPS'///)
      GOTO 9000
3333  CONTINUE
c
      OPEN(4,FILE=FILE2,STATUS='NEW')
8000  FORMAT(' ANALYSIS OUTPUT FROM DATA FILE: ',A15/)
4     FORMAT(' START/INSP  END/INSP TIDAL VOLUME START/EXP  END/EXP')
c
c     Routine for writing spirometric volumes
c
      IF (A.NE.3) GOTO 3009
      WRITE(*,3337)
      DO 3339 P=1,I
      WRITE(*,3338) P, TIDALV(P)
3339  CONTINUE
3337  FORMAT('  VOLUME #   SPIROMETRIC VOLUMES (inspiratory)')
3338  FORMAT(2X,I4,11X,F7.3)
      GOTO 9000
3009  CONTINUE
c
      DO 40 P=1,(I-1)
c     WRITE(4,41) IONSET(P),EONSET(P),TIDALV(P),STARTEXP(P),ENDEXP(P)
c     WRITE(*,41) IONSET(P),EONSET(P),TIDALV(P),STARTEXP(P),ENDEXP(P)
c40   CONTINUE
41    FORMAT(15(I6,7X,I6,3X,F12.4,2X,I6,4X,I6))
c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
c       RESPIRATORY TIMING SUBROUTINES (including post-insp
c       [POSTINSP] and post-expiratory pauses [POSTEXP]).
c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
      DO 42 P=1,(I-1)
      TINSP(P) = (EONSET(P) - IONSET(P)) / 50.0
      TEXP(P) = (ENDEXP(P)-STARTEXP(P))/50.0
      TTOTAL(P) = (IONSET(P+1)-IONSET(P))/50.0
      POSTINSP(P) = (STARTEXP(P)-EONSET(P))/50.0
      POSTEXP(P) = (IONSET(P+1)-ENDEXP(P))/50.0
42    CONTINUE
c
      WRITE(*,103) I,E
103   FORMAT('    ',I5,' INSPIRATIONS AND ',I3,' EXPIRATIONS'///)
```

```
      WRITE(4,420)
      WRITE(*,420)
420   FORMAT(' TIDAL/I TIDAL/E INSP/Time EXP/Time T/tot '
     *  ' POST-INSP/Pause POST-EXP/Pause')
c
      DO 421 P=1,(I-1)
      WRITE(*,422) TIDALV(P),EXTIDALV(P),TINSP(P),TEXP(P),TTOTAL(P),
     *  POSTINSP(P),POSTEXP(P)
      WRITE(4,422) TIDALV(P),EXTIDALV(P),TINSP(P),TEXP(P),TTOTAL(P),
     *  POSTINSP(P),POSTEXP(P)
421   CONTINUE
422   FORMAT(F7.3,2X,F7.3,2X,F7.4,1X,F7.4,1X,F7.4,5X,F7.4,8X,F7.4)
c
      AVTINSP=0.0
      AVTEXP=0.0
      AVTTOTAL=0.0
      AVTIDALV = 0.0
      AVINSPPOST = 0.0
      AVEXPPOST = 0.0
      DO 44 P=1,(I-1)
      AVTIDALV = AVTIDALV + TIDALV(P)
       AVTINSP = AVTINSP + TINSP(P)
      AVTEXP = AVTEXP + TEXP(P)
      AVTTOTAL = AVTTOTAL + TTOTAL(P)
      AVINSPPOST = AVINSPPOST + POSTINSP(P)
      AVEXPPOST = AVEXPPOST + POSTEXP(P)
44    CONTINUE
c
      AVTIDALV = AVTIDALV / (I-1)
      AVTINSP = AVTINSP / (I-1)
      AVTEXP = AVTEXP / (I-1)
      AVTTOTAL = AVTTOTAL / (I-1)
      RFREQ=60.0/AVTTOTAL
      VTTIMEINSP = AVTIDALV / AVTINSP
      MINVENT = AVTIDALV * RFREQ
      TINTOTAL = AVTINSP / AVTTOTAL
      AVINSPPOST = AVINSPPOST / (I-1)
      AVEXPPOST = AVEXPPOST / (I-1)
c     MINV2 = VTTIMEINSP * TINTOTAL
c
      WRITE(*,440) AVTIDALV
      WRITE(4,440) AVTIDALV
440   FORMAT(//////' AVERAGE TIDAL VOLUME = ',F12.4/)
      WRITE(*,441) AVTINSP
      WRITE(4,441) AVTINSP
441   FORMAT(' AVERAGE TIME INSP = ',F12.4)
      WRITE(*,442) AVTEXP
      WRITE(4,442) AVTEXP
442   FORMAT(' AVERAGE TIME EXPIR. = ',F12.4)
      WRITE(*,443) AVTTOTAL
      WRITE(4,443) AVTTOTAL
443   FORMAT(' AVERAGE TOTAL TIME = ',F12.4/)
```

```
      WRITE(*,444) VTTIMEINSP
      WRITE(4,444) VTTIMEINSP
444   FORMAT('   VT/Ti =   ',F12.4)
      WRITE(*,446) TINTOTAL
      WRITE(4,446) TINTOTAL
      WRITE(*,4400) RFREQ
      WRITE(4,4400) RFREQ
4400  FORMAT(' BREATHING FREQUENCY = ',F12.4)
      WRITE(*,445) MINVENT
      WRITE(4,445) MINVENT
445   FORMAT(' MINUTE VENT. = ',F12.4/)
446   FORMAT(' Ti/Ttot = ',F12.4/)
      WRITE(*,447) AVINSPPOST
      WRITE(4,447) AVINSPPOST
447   FORMAT(' AVERAGE POST-INSPIRATORY PAUSE = ',F12.4)
      WRITE(*,448) AVEXPPOST
      WRITE(4,448) AVEXPPOST
448   FORMAT(' AVERAGE POST-EXPIRATORY PAUSE = ',F12.4)
c
      CLOSE(4)
      GOTO 9000


ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
c     COMPUTING:  FLOW-VOLUME RELATIONSHIPS.
c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
6000  OPEN(4,FILE=FILE2,STATUS='NEW')
      TOPVOL=0.0
c
    DO 6 P=1,I
c
       DO 60 J=1,200
         RVOLIN(J)=0.0
           VOLEX(J)=0.0
         RINFLOW(J)=0.0
         RVOLEX(J)=0.0
         REXFLOW(J)=0.0
60      CONTINUE
c
           WRITE(*,63)
c          WRITE(4,63)
       DO 610 PPP=IONSET(P),(EONSET(P)-1)
         PP=(PPP-IONSET(P))+1
         INVOL=PNEUMO(PPP)-BASEFLOW
         RINVOL=FLOAT(INVOL)
         RINVOL=RINVOL/VOLAVE*(PBAR+PREG)/(PBAR)*310/
     *     (273+TA)*(PBAR+PLC-PH2O)/(PBAR+PLC-64.056)
         IF (RINVOL.LT.(0.0)) GOTO 6100
         RINVOL=0.0-RINVOL
```

```
6100          RVOLIN(PP)=RVOLIN(PP-1)-RINVOL
              RINFLOW(PP)=FLOAT(INVOL)
              RINFLOW(PP)=RINFLOW(PP)/VOLAVE*50.0*(PBAR+PREG)/(PBAR)
     *          *310/(273+TA)*(PBAR+PLC-PH2O)/(PBAR+PLC-64.056)
              WRITE(*,631) PPP,RVOLIN(PP),RINFLOW(PP)
              WRITE(4,631) PPP,RVOLIN(PP),RINFLOW(PP)
610           CONTINUE
c
                 TOPVOL=RVOLIN(PP)
c
               WRITE(*,630)
c              WRITE(4,630)
c
            DO 620 PPP=STARTEXP(P),(ENDEXP(P)-1)
              PP=(PPP-STARTEXP(P))+1
              EXVOL=PNEUMO(PPP)-BASEFLOW
              REXVOL=FLOAT(EXVOL)
              REXVOL=REXVOL/VOLAVE*(PBAR+PREG)/(PBAR)*310/
     *          (273+TA2)*(PBAR+PLC-PH2O2)/(PBAR+PLC-64.056)
              RVOLEX(PP)=RVOLEX(PP-1)+REXVOL
                 VOLEX(PP)=TOPVOL-RVOLEX(PP)
              REXFLOW(PP)=FLOAT(EXVOL)
              REXFLOW(PP)=REXFLOW(PP)/VOLAVE*50.0*(PBAR+PREG)/(PBAR)
     *          *310/(273+TA2)*(PBAR+PLC-PH2O2)/(PBAR+PLC-64.056)
              WRITE(*,631) PPP,VOLEX(PP),REXFLOW(PP)
              WRITE(4,631) PPP,VOLEX(PP),REXFLOW(PP)
620           CONTINUE
c
6             CONTINUE
c
63            FORMAT(' INSPIRATORY DATA'/
     *        ' #  Vol(BTPS)  Flow  ')
630           FORMAT(/' EXPIRATORY DATA'/
     *        ' #  Vol(BTPS)  Flow  ')
631           FORMAT(150(I4,1X,F8.4,2X,F9.4))
              CLOSE(4)
c
c  Subroutines for computing FEV1.0.
c
              EVOL=0
              FEV1=0.0
              EXVOLINSP=0.0
C
              DO 64 P=STARTEXP(1),(STARTEXP(1)+49)
                    EVOL=PNEUMO(P)-BASEFLOW
                    ERVOL=FLOAT(EVOL)
                    EXVOLINSP=EXVOLINSP+ERVOL
64            CONTINUE
c
              FEV1=EXVOLINSP/VOLAVE*(PBAR+PREG)/(PBAR)*310
     *       /(273+TA2)*(PBAR+PLC-PH2O2)/(PBAR+PLC-64.056)
c
```

```
      WRITE(*,65) FEV1
65    FORMAT(///'  FEV1 = ',F8.4,' Litres BTPS.'///)
c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
9000  WRITE(*,9996) FILE1
9996  FORMAT(/' ANALYSIS OF ',A20,' IS COMPLETE.'/)
      WRITE(*,9997)
9997  FORMAT(' WHAT DO YOU WISH TO DO NEXT?'/
     *  ' 1 RETURN = ANALYSE ANOTHER FILE WITH SAME CALIBRATION,'/
     *  ' 9 RETURN = END Program.'/)
      READ(*,100) REPEAT
      IF (REPEAT.EQ.1) GOTO 1310
      IF (REPEAT.EQ.9) CONTINUE
c
9999  STOP
      END
```

## APPENDIX 8

Example analysis program for use during an exercise protocol.

Many of the comments necessary to understand this program and its variables have been omitted where they have been provided in preceeding Appendices.

```
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
c  Program TO ANALYSE RESPIRATORY VOLUME AND MOUTH PRESSURE DATA.
c
c        Program uses data obtained from A/D program
c        using the DASH16 A/D board (MetraByte, at 50Hz),
c        and produces breath-by-breath and 20 sec average output.
c
c  ASCII INPUT DATA REQUIRED:
c             ASCII file contains 2 columns of data:
c             Column 1 = mouth pressure, Column 2 = flow.
c
c  THE FOLLOWING ANALYSES ARE PERFORMED:
c    (1) Tidal volumes, (2) Respiratory timing (Ti, Te, Ttot,
c    Vt/Ti, Ti/Ttot, fb), (3) Minute ventilation, (4) Peak-to-peak
c    mouth pressures (cmH2O), and (5) flow-volume-pressure data
c    for the first breath of each 20 sec interval.
c
c Limitations: (1) Expiratory data cannot be measured using this
c                program without changes. It has been designed
c                for a set-up using inspiratory pneumotach only.
c             (2) Data is read in from an ASCII file input only.
c             (3) Convention for flow: inspiration is negative.
c
c  NIGEL A.S. TAYLOR.              June 1990
c     Diving Medicine, Naval Medical Research Institute
c
c  LATEST UPDATE: 10/12/90
c
c    This program allows for unequal inspired/expired volumes
c    by using only complete cycles, starting with an inspiration.
c    Inspiration is detected from a negative signal from the flow
c    channel. The phase shift between the flow and the pressure
c    signals is taken into account during analysis.
c
c        Three types of data output files are used:
c  (1) file holding breath-by-breath data (variables 1 to 4);
c  (2) file holding flow-volume-pressure data (20 msec intervals);
c  (3) a database holding only mean values from variables in file 1.
c  These correspond with Units 7, 6 and 3 respectively in program.
c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
$STORAGE:2
```

```
      INTEGER ZEROFLOW,LITER,BASELIM,PHASE1,PHASE2,INVOL,ZZZ,EPRESS
      INTEGER PNEUMO,NOFLOW,FLOW,BASE,IDATA,COMP,EXVOL,CODE,IPRESS
      INTEGER GAIN,IONSET,EONSET,SEC15,POINT,POINT2,EXPER,LIMBASE
      INTEGER MAX2,IVOL,BASEFLOW,TBASELIM,EX,EXP,XX,PLOOP,MLOOP
      INTEGER A,B,C,D,E,F,I,J,K,L,M,N,P,PP,PPP,Q,R,S,T,W,X,Y,Z
      INTEGER ARBITLIM,PRESS,COUNT,ECOUNT,A2,TIME,LIMITVOL,ELIMBASE
C
      REAL VOLUME,VOLAVE,RVOLSUM,PBAR,PH2O,TA,AVTIDALV,TINSP,ETIME
      REAL TOTALVOL,INSPVOL,SYRINGE,VOL,TIDALV,TEXP,TTOTAL,AVRATIO
      REAL VOLINSP,CALVOL,RVOL,VTTIMEINSP,AVTINSP,REGCOEFF,CONSTANT
      REAL AVTEXP,AVTTOTAL,RFREQ,MINVENT,MINV2,TINTOTAL,RSEC15,ZERO
      REAL SMALLTV,TIDE,REXVOL,VOLEX,EXT5,MAX,AVEXPPOST,TOPVOL,RVOLEX
      REAL RINVOL,RVOLIN,RINFLOW,REXFLOW,PINSP,PEXP,INT,EXT,INT5
      REAL A3,A4,A5,A6,A7,A8,A9,A10,A11,A12,A13,RATIO,EBIT,BIT
      REAL*4 PEAK,EPEAK,MPPEAK,PPEAK,MPRESS,EMPRESS
C
      DIMENSION IDATA(2,1000),LITER(2000),PRESS(1000),MPRESS(900)
      DIMENSION PNEUMO(1000),VOL(15),VOLBTPS(15),IONSET(15),TINSP(15)
      DIMENSION CALVOL(9),EONSET(15),TIDALV(15),TEXP(15),TTOTAL(15)
      DIMENSION PINSP(15),PEXP(15),EMPRESS(900)
      DIMENSION RVOLIN(250),RINFLOW(250),RATIO(15)
      DIMENSION PEAK(900),EPEAK(900),PPEAK(15)
C
      CHARACTER*10 FILENAME,FILE1,FILE2,FILE4,FILE5,FILE6,FILE7
      CHARACTER*12 FILE3,ID,A1
C
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
C
C     INTEGRATION OF FLOW TO GET TIDAL VOLUMES.
C
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      OPEN(3)
      WRITE(*,1)
1     FORMAT(/////////////////
     *  '              GERMAN 1500 fsw DIVE SERIES:'//
     *  '         RESPIRATORY VOLUME AND MOUTH PRESSURE'/
     *  '                ANALYSIS SOFTWARE.'///
     *  '  '///////)
C
      FLOW=-200
      SYRINGE = 3.000
      ARBITLIM = 4000
      FILE3 = 'GERMBASE.PRN'
C
C
100   FORMAT(I1)
1006  WRITE(*,1000)
1000  FORMAT(//'  ENTER NAME OF ZERO FLOW CALIBRATION FILE:'//)
      READ(*,20) FILE1
      WRITE(*,1001)
1001  FORMAT(//'  ENTER NAME OF HIGH FLOW CALIBRATION FILE:'//)
      READ(*,20) FILE2
```

```
      WRITE(*,1002)
1002  FORMAT(//' ENTER NAME OF PRESSURE CALIBRATION FILE:'//)
      READ(*,20) FILE4
C
C     Reading calibration data from files.
C
      OPEN (6,FILE=FILE4,STATUS='OLD')
      READ(6,50) REGCOEFF,CONSTANT
      CLOSE(6)
      WRITE(*,5099) REGCOEFF,CONSTANT
5099  FORMAT(/' REGRESSION COEFFICIENT (cmH2O) = ',F9.2/
     *  '   CONSTANT A (cmH2O) = ',F9.2//)
c
      OPEN(3,FILE=FILE1,STATUS='OLD')
      READ(3,101) ZEROFLOW
      CLOSE(3)
c
      WRITE(*,102) ZEROFLOW
101   FORMAT(I7)
102   FORMAT(' BASAL FLOW (RAW UNITS) = ',I5//)
50    FORMAT(2X,F9.2,2X,F9.2)
c
      BASE = (FLOW - ZEROFLOW) * 4 / 100
      BASELIM = BASE + ZEROFLOW
      TBASELIM = ZEROFLOW - BASE
C     WRITE(*,103) BASE
103   FORMAT(' VALUE OF BASE = 'I5//)
C     WRITE(*,104) BASELIM
104   FORMAT(' VALUE OF BASELIMIT = 'I5//)
C
      OPEN(3,FILE=FILE2,STATUS='OLD',FORM='UNFORMATTED')
      READ(3) (LITER(J),J=1,2000)
11    FORMAT(10(I7))
      CLOSE(3)
c
      M=1
      L=0
12    DO 120 J=M,2000
      IF (LITER(J).LT.BASELIM) GOTO 1200
120   CONTINUE
1200  CONTINUE
      IF (J.GE.2000) GOTO 129
c                                        Integration of flow data.
      VOLINSP=0.0
121   DO 127 T=J,2000
      IVOL=LITER(T)-ZEROFLOW
      RVOL=FLOAT(IVOL)
      VOLINSP=VOLINSP-RVOL
      IF (LITER(T).GE.BASELIM) GOTO 1270
127   CONTINUE
1270  CONTINUE
c                                  Checking data for false volumes before storing volumes.
```

```
      IF (VOLINSP.LE.ARBITLIM) GOTO 128
      L=L+1
      CALVOL(L)=VOLINSP
C     WRITE(*,1391) L,CALVOL(L)
1391  FORMAT('  FOR PUMP ',I3,' CALVOL = ',F12.1)
128   IF (T.GE.2000) GOTO 129
      M=T
      GOTO 12
c                                   Calculation of average calibration value.
129   RVOLSUM=0.0
      DO 1280 W=1,L
      RVOLSUM=RVOLSUM+CALVOL(W)
1280  CONTINUE
      VOLAVE = RVOLSUM / (SYRINGE*L)
1281  FORMAT(10(I7))
      WRITE(*,1282) VOLAVE
1282  FORMAT(/' MEAN FLOW CALIBRATION VALUE =',F14.4,' bits/litre'//)
      WRITE(*,1360)
1360  FORMAT(//'          LOOK AT THE VOLUME CALIBRATION VALUE:'/
     *  '    SELECT A FALSE VOLUME LIMIT BASED UPON THIS VALUE:'/
     *  '              [e.g. 200 ml = 20%]'//
     *  '         ENTER YOUR VALUE {I7 - NO decimal}'//)
      READ(*,1361) LIMITVOL
1361  FORMAT(I9)
C
2     FORMAT(//'  ENTER NAME OF DATA FILE TO BE ANALYSED:'///)
20    FORMAT(A10)
14    FORMAT(///'  HOW MANY MINUTES DID THIS EXPERIMENT RUN? [I3]'//)
      WRITE(*,130)
130   FORMAT(/' WHAT WAS THE BAROMETRIC PRESSURE (mmHg F12.2)?'/)
      READ(*,131) PBAR
131   FORMAT(F12.2)
      PBAR = PBAR * 1.36
      WRITE(*,132)
132   FORMAT(//' WHAT WAS THE INSPIRED AIR TEMPERATURE (degC F8.4)?'/)
      READ(*,131) TA
      WRITE(*,133)
133   FORMAT(//' WHAT WAS INSPIRED WATER VAPOUR PRESSURE (mmHg F8.4)?'/)
      READ(*,131) PH2O
      PH2O = PH2O * 1.36
c


      COMP = 0
      COMPRESS= 0.0
      PLC = 0
      SIPLC = 0
      PREG = 0
      PSTERN = 0
      PHASE1 = 0
c             PHASE1 comment:
c  The flow and mouth pressure signals are rarely in phase so
c  this difference must be taken into account. Let the value for
```

```
c  the flow signal be zero, then subtract the number of points
c  between the 2 signals from zero. Report the sign below.
c

       WRITE(*,8905)
8905   FORMAT(//' WHAT WAS THE PHASE SHIFT BETWEEN FLOW AND MOUTH'
     *  '       PRESSURE [FLOW MINUS PRESSURE: I3]?'//)
       READ(*,135) PHASE1
C
135    FORMAT(I3)
1444   FORMAT(I2)
1310   WRITE(*,2)
       READ(*,20) FILENAME
       WRITE(*,14)
       READ(*,1444) SEC15
       WRITE(*,8901)
8901   FORMAT(//' ENTER SUBJECT'S ID CODE:'//)
       READ(*,20) ID
       WRITE(*,8904)
8904   FORMAT(//' ENTER CONDITION CODE (numerical I3):'//)
       READ(*,101) CODE
       WRITE(*,8902)
8902   FORMAT(//' ENTER NAME OF FILE FOR FLOW-VOL-PRESSURE OUTPUT:'//)
       READ(*,20) FILE6
       WRITE(*,8903)
8903   FORMAT(//' ENTER NAME OF FILE FOR BREATH-BY-BREATH OUTPUT:'//)
       READ(*,20) FILE7
C
c      Reading formatted file.
c Placing flow data into the array PNEUMO and pressure data into PRESS
C
       OPEN(3,FILE=FILENAME,STATUS='OLD',FORM='FORMATTED')
       OPEN(5,FILE=FILE3,STATUS='OLD',FORM='FORMATTED')
       DO 1592 B=1,10000
       READ(5,1590,END=1591) A1,A2,A3,A4,A5,A6,A7,A8,A9,A10,A11,A12,A13
1592   CONTINUE
1591   CONTINUE
1590   FORMAT(A6,I5,11F8.3)
       ENDFILE 5
       BACKSPACE 5
       OPEN(6,FILE=FILE6,STATUS='NEW',FORM='FORMATTED')
       OPEN(7,FILE=FILE7,STATUS='NEW',FORM='FORMATTED')
c
       TIME = 0
       ETIME = 0.0
       PLOOP = 0
       SEC15 = SEC15 * 3
       MLOOP = SEC15
c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
       DO 4446 PLOOP = 1,MLOOP
```

```
c
                ETIME = ETIME + 0.3334
                TIME = TIME + 1
                POINT = 1
                POINT2 = 1000
                BASEFLOW=ZEROFLOW
                J=0
                K=0
c
        DO 21 K=POINT,POINT2
        READ(3,2090,END=2999) (IDATA(L,K),L=1,2)
21      CONTINUE
        GOTO 2998
2090    FORMAT(2I6)
2999    WRITE(*,24) FILENAME
2998    CONTINUE
C
        DO 22 J=POINT,POINT2
        PRESS(J)=IDATA(1,J)
        PNEUMO(J)=IDATA(2,J)
22      CONTINUE
C
C       WRITE(*,2292) IDATA(1,POINT)
2292    FORMAT('  STARTING POINT FOR 20 SEC ',I5)
C       WRITE(*,2293) J
2293    FORMAT(' FINISHING POINT FOR 20 SEC ',I4)
C
        GOTO 6003
C
24      FORMAT('                    END OF FILE ENCOUNTERED.'/
     *  '          ANALYSIS OF FILE: ',A10,' COMPLETE.'//)
c
c           Volume analysis routine.
6003    N=0


c
        DO 3002 J=1,15
          TIDALV(J)=0.0
          VOLBTPS(J)=0.0
          IONSET(J)=0
          EONSET(J)=0
          TINSP(J)=0.0
          TEXP(J)=0.0
          TTOTAL(J)=0.0
3002    CONTINUE
        TOTALVOL = 0.0
c
c  COUNTERS: I = inspiration onset, E = expiration onset,
c          P = pairs of insp and exp, J,K,M = starting points,
c          L = tidal volumes.
c  NOTE: I is used as a respiratory cycle counter from this point.
```

```
c  Integrating flow to obtain volume (inspiration -ive).
c
        E=0
              I=0
              J=0
              K=0
              L=0
        M=1
C
3       CONTINUE
C
        DO 3111 J=M,POINT2
        IF (PNEUMO(J).GE.BASELIM) GOTO 3112
C       WRITE(*,8820) PNEUMO(J),J
3111    CONTINUE
88220   FORMAT('+ PNEUMO(J) = ',I6,' FOR J = ',I6)
C       WRITE(*,8822)
8822    FORMAT(//'                FINISHED LOOP '//)
C
3112    DO 31 K=J,POINT2
        IF (PNEUMO(K).LT.BASELIM) GOTO 310
C       WRITE(*,8820) PNEUMO(J),J
31      CONTINUE
C
310     CONTINUE
C       WRITE(*,8822)
        IF (K.GE.POINT2) GOTO 3102
        VOLINSP=0.0
        EXVOLINSP=0.0
c
c
c
c
c
        DO 32 T=K,POINT2
              IF (PNEUMO(T).GE.BASELIM) GOTO 33
              IVOL=PNEUMO(T)-BASEFLOW
              RVOL=FLOAT(IVOL)
              VOLINSP = VOLINSP - RVOL
32      CONTINUE
33      CONTINUE
c                     The -RVOL converts volume to positive.
c                           Check for small, false volumes.
c
        IF (VOLINSP.LT.ARBITLIM) GOTO 3103
c
c  Marking points of inspiration start (IONSET), & end (EONSET),
c    and expiration start (STARTEXP), & end (ENDEXP).
c
        I=I+1
        IONSET(I)=K
        EONSET(I)=T
```

```
C
      DO 330 J=1,15
      VOL(J)=0.0
330   CONTINUE
c
      L=L+1
3390        FORMAT('       VOLUME INSPIRED = ',F6.3)
            VOL(L) = VOLINSP / VOLAVE
C     WRITE(*,3390) VOL(L)
c
c               Volume BTPS correction (deg C and cmH2O units)
c
      VOL(L) = VOL(L) * (PBAR+PREG) / (PBAR)
      VOLBTPS(L)=VOL(L)*310/(273+TA)*(PBAR+PLC-PH2O)
     *          /(PBAR+PLC-64.056)
      TIDALV(L)=VOLBTPS(L)
4199        FORMAT(3I6,F8.3)
c
3103  IF (T.GE.POINT2) GOTO 3102
C           WRITE(*,4199) L,IONSET(L),EONSET(L),TIDALV(L)
      IF (VOLINSP.GT.ARBITLIM) GOTO 3104
      M = T
      GOTO 3
c
3104        CONTINUE
      E=E+1
3410  CONTINUE
      M=T
      GOTO 3
3102  CONTINUE
c
8810        FORMAT(' HERE',I5)
8000  FORMAT(' ANALYSIS OUTPUT FROM DATA FILE: ',A15/)
4     FORMAT(' START/INSP  END/INSP TIDAL VOLUME')
41          FORMAT(3I6,3X,2F12.4)
c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
c       RESPIRATORY TIMING SUBROUTINES (including post-insp
c       [POSTINSP] and post-expiratory pauses [POSTEXP]).
c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
            IF (I.GT.1) ZZZ = (I-1)
            IF (I.LE.1) ZZZ = 1
c
      DO 42 P=1,ZZZ
       TINSP(P) = (EONSET(P) - IONSET(P)) / 50.0
      IF((I-1).LE.0) GOTO 4271
       TTOTAL(P) = (IONSET(P+1)-IONSET(P))/50.0
       IF (TTOTAL(P).LE.(0.0)) GOTO 4271
       GOTO 4272
```

```
4271    TTOTAL(P) = TINSP(P) * 2
4272    TEXP(P) = TTOTAL(P) - TINSP(P)
42      CONTINUE
C
422     FORMAT(15(F7.4,5X,F7.4,1X,F7.4))
C
        AVTINSP=0.0
        AVTEXP=0.0
        AVTTOTAL=0.0
        AVTIDALV = 0.0
c
        DO 44 P=1,ZZZ
        IF ((I-1).LE.0) GOTO 4007
        AVTIDALV = AVTIDALV + TIDALV(P)
        AVTINSP = AVTINSP + TINSP(P)
        AVTEXP = AVTEXP + TEXP(P)
        AVTTOTAL = AVTTOTAL + TTOTAL(P)
44    CONTINUE
C
        AVTIDALV = AVTIDALV / (I-1)
        AVTINSP = AVTINSP / (I-1)
        AVTEXP = AVTEXP / (I-1)
        AVTTOTAL = AVTTOTAL / (I-1)
              IF (AVTTOTAL.LE.(0.0).OR.AVTINSP.LE.(0.0)) GOTO 4009
        RFREQ=60.0/AVTTOTAL
        VTTIMEINSP = AVTIDALV / AVTINSP
        MINVENT = AVTIDALV * RFREQ
        TINTOTAL = AVTINSP / AVTTOTAL
              GOTO 4008
4007    AVTIDALV = TIDALV(1)
        AVTINSP = TINSP(1)
        AVTEXP = TEXP(1)
        AVTTOTAL = TTOTAL(1)
        IF (TTOTAL(1).LE.(0.0).OR.TINSP(1).LE.(0.0)) GOTO 4009
        RFREQ = 60.0 / TTOTAL(1)
        VTTIMEINSP = TIDALV(1) / TINSP(1)
        MINVENT = TIDALV(1) * RFREQ
        TINTOTAL = TINSP(1) / TTOTAL(1)
        GOTO 4008
4009    RFREQ = RFREQ
        VTTIMEINSP = VTTIMEINSP
        MINVENT = TIDALV(1) * RFREQ
        TINTOTAL = TINTOTAL
4008    CONTINUE
c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
c           FLOW-VOLUME-PRESSURE SUB-ROUTINES
c Inspiratory output only, for the first and last breaths and for
c the first breath in the middle 20 sec period of each 2 minutes.
c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
```

```
c
      TOPVOL=0.0
      BIT = 0.0
      EBIT = 0.0
c
      DO 500 P=1,ZZZ
c
            IF (P.NE.1) GOTO 500
c
                  DO 501 J=1,250
                  RVOLIN(J)=0.0
                  RINFLOW(J)=0.0
501         CONTINUE
                  DO 5011 J=1,900
                  MPRESS(J)=0.0
                  PPEAK(J)=0.0
                  EMPRESS(J)=0.0
                  EPEAK(J)=0.0
5011              CONTINUE
c



      DO 510 PPP=IONSET(P),(EONSET(P)-1)
            PP=(PPP-IONSET(P))+1
            BIT = BIT + 0.02
            IPRESS=PRESS(PPP+PHASE1)
            MPRESS(PP)=FLOAT(IPRESS)
            MPRESS(PP)=(MPRESS(PP)-CONSTANT)/REGCOEFF
            INVOL=PNEUMO(PPP)-BASEFLOW
            RINVOL=FLOAT(INVOL)
            RINVOL=RINVOL/VOLAVE*(PBAR+PREG)/(PBAR)*310/
     *      (273+TA)*(PBAR+PLC-PH2O)/(PBAR+PLC-64.056)
            IF (RINVOL.LT.(0.0)) GOTO 5100
            RINVOL=0.0-RINVOL
5100        RVOLIN(PP)=RVOLIN(PP-1)-RINVOL
            RINFLOW(PP)=FLOAT(INVOL)
            RINFLOW(PP)=RINFLOW(PP)/VOLAVE*50.0*(PBAR+PREG)/(PBAR)
     *      *310/(273+TA)*(PBAR+PLC-PH2O)/(PBAR+PLC-64.056)
            IF (TIME.EQ.1) GOTO 5110
            IF (TIME.EQ.5) GOTO 5110
            IF (TIME.EQ.11) GOTO 5110
            IF (TIME.EQ.17) GOTO 5110
            IF (TIME.EQ.23) GOTO 5110
            IF (TIME.EQ.29) GOTO 5110
            IF (TIME.EQ.35) GOTO 5110
            IF (TIME.EQ.41) GOTO 5110
            IF (TIME.EQ.47) GOTO 5110
            IF (TIME.EQ.53) GOTO 5110
            IF (TIME.EQ.59) GOTO 5110
```

```
              IF (TIME.EQ.65) GOTO 5110
              IF (TIME.EQ.71) GOTO 5110
              IF (TIME.EQ.75) GOTO 5110
              GOTO 5111
5110          CONTINUE
c             WRITE(*,531) BIT,RVOLIN(PP),MPRESS(PP),RINFLOW(PP)
              WRITE(6,531) BIT,RVOLIN(PP),MPRESS(PP),RINFLOW(PP)
5111          CONTINUE
510     CONTINUE
c
c       TOPVOL=RVOLIN(PP)
c
c       DO 520 PPP=STARTEXP(P),(ENDEXP(P)-1)
c             PP=(PPP-STARTEXP(P))+1
c             EBIT = EBIT + 0.02
c             IPRESS=PRESS(PPP+PHASE1)
c             EMPRESS(PP)=FLOAT(IPRESS)
c             EMPRESS(PP)=(EMPRESS(PP)-CONSTANT)/REGCOEFF
c             EXVOL=PNEUMO(PPP)-BASEFLOW
c             REXVOL=FLOAT(EXVOL)
c             REXVOL=REXVOL/VOLAVE*(PBAR+PREG)/(PBAR)*310/
c     *       (273+TA2)*(PBAR+PLC-PH2O2)/(PBAR+PLC-64.056)
c             RVOLEX(PP)=RVOLEX(PP-1)+REXVOL
c             VOLEX(PP)=TOPVOL-RVOLEX(PP)
c             REXFLOW(PP)=REXFLOW(PP)/VOLAVE*50.0*(PBAR+PREG)/(PBAR)
c     *       *310/(273+TA2)*(PBAR+PLC-PH2O2)/(PBAR+PLC-64.056)
c             WRITE(*,531) EBIT,VOLEX(PP),EMPRESS(PP),REXFLOW(PP)
c             WRITE(6,531) EBIT,VOLEX(PP),EMPRESS(PP),REXFLOW(PP)
520     CONTINUE
c
500     CONTINUE

531     FORMAT(F8.3,3X,F8.3,2X,F8.3,2X,F9.3)
c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
c             MOUTH PRESSURE CALCULATIONS
c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
              COUNT=1
              ECOUNT=1
              MPPEAK=0.0
              AVRATIO=0.0
              ZERO=CONSTANT
c
              DO 60 J=1,900
                    MPRESS(J)=0.0
                    EMPRESS(J)=0.0
                    PEAK(J)=0.0
                    EPEAK(J)=0.0
60            CONTINUE
```

```
            DO 6111 J=1,15
                  PPEAK(J)=0.0
                  RATIO(J)=0.0
6111        CONTINUE
c
      DO 6 P=1,ZZZ
c
            DO 600 PPP=IONSET(P),(EONSET(P)-1)
                  PP=(PPP-IONSET(P))+1
                  IPRESS=PRESS(PPP+PHASE1)
                  MPRESS(PP)=FLOAT(IPRESS)
                  MPRESS(PP)=(MPRESS(PP)-CONSTANT)/REGCOEFF
                  IF (MPRESS(PP).GE.PEAK(COUNT)) GOTO 6000
                  COUNT = COUNT + 1
                  PEAK(COUNT) = MPRESS(PP)
c                 WRITE(*,6401) PEAK(COUNT)
6000              CONTINUE
600         CONTINUE
c
c           IF (PEAK(COUNT).LT.(-100.0)) PEAK(COUNT)=0.0
c           WRITE(*,6400) PEAK(COUNT)
6400        FORMAT('          PEAK INSPIRATORY PRESSURE = ',F9.3)
6404        FORMAT('          PEAK EXPIRATORY PRESSURE = ',F9.3)
6401        FORMAT('+  INSPIRATORY PRESSURE REAL =',F9.1)
6403        FORMAT('+  EXPIRATORY PRESSURE REAL =',F9.1)
6405  FORMAT(' PEAK-TO-PEAK= 'F9.1' MEAN='F9.1' BREATH ',I3/)
c
            DO 610 PPP=EONSET(P),(IONSET(P+1)-1)
                  PP=(PPP-EONSET(P))+1
                  EPRESS=PRESS(PPP+PHASE1)
                  EMPRESS(PP)=FLOAT(EPRESS)
                  EMPRESS(PP)=(EMPRESS(PP)-CONSTANT)/REGCOEFF
                  IF (EMPRESS(PP).LE.EPEAK(ECOUNT)) GOTO 6100
                  ECOUNT = ECOUNT + 1
                  EPEAK(ECOUNT) = EMPRESS(PP)
c                 WRITE(*,6403) EPEAK(ECOUNT)
6100              CONTINUE
610         CONTINUE
c
c           IF (EPEAK(ECOUNT).GT.(100.0)) EPEAK(ECOUNT)=0.0
c           WRITE(*,6404) EPEAK(ECOUNT)
c
            PPEAK(P)=EPEAK(ECOUNT)-PEAK(COUNT)
c           WRITE(*,6402) PPEAK(P),P
6402        FORMAT('  PEAK-TO-PEAK PRESSURE = ',F9.3,' BREATH'I4)
            IF (EPEAK(ECOUNT).LE.(0.0)) GOTO 6138
            RATIO(P)=(0.0-PEAK(COUNT))/EPEAK(ECOUNT)
            GOTO 6139
6138        RATIO(P)=0.0
6139        CONTINUE
c
            WRITE(7,41) P,IONSET(P),EONSET(P),TIDALV(P),PPEAK(P)
```

```
c               IF ((I-1).LE.0) GOTO 6288
                MPPEAK = MPPEAK + PPEAK(P)
C               WRITE(*,6405) PPEAK(P),MPPEAK,P
                AVRATIO = AVRATIO + RATIO(P)
                GOTO 6289
6288            CONTINUE
                MPPEAK = PPEAK(1)
                AVRATIO = RATIO(1)
6289            CONTINUE
c
6       CONTINUE
c
                MPPEAK = MPPEAK / ZZZ
                AVRATIO = AVRATIO / ZZZ
c
c
                IF (PLOOP.GT.1) GOTO 4445
c
                WRITE(*,4444)
4444            FORMAT(//
     *      '                AVERAGE VALUES IN 20 SECOND SEGMENTS'/
     *      '   Vt   P-Peak    Ti     Te     Ttot    Vt/Ti '
     *      '  fb    Ve   Ti/Ttot')
c
4445            WRITE(*,1599) AVTIDALV,MPPEAK,
     *          AVTINSP,AVTEXP,AVTTOTAL,VTTIMEINSP,RFREQ,
     *          MINVENT,TINTOTAL
1599            FORMAT(9F8.3)
                WRITE(5,1590) ID,CODE,ETIME,AVTIDALV,MPPEAK,AVRATIO,
     *          AVTINSP,AVTEXP,AVTTOTAL,VTTIMEINSP,RFREQ,
     *          MINVENT,TINTOTAL
c
c
4446  CONTINUE
c
c
ccccccccccccccccccccccc cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
9995    WRITE(*,9996) FILENAME
9996    FORMAT(///' ANALYSIS OF FILE ',A10,' IS COMPLETE.'///)
        CLOSE(3)
        CLOSE(5)
        CLOSE(6)
        CLOSE(7)
        A=0
        WRITE(*,9997)
9997  FORMAT(' WHAT DO YOU WISH TO DO NEXT?'/
     *    ' 1 RETURN = ANALYSE MORE DATA USING SAME CAL. FILES,'/
     *    ' 6 RETURN = ANALYSE MORE DATA USING DIFFERENT CAL. FILES,'/
     *    ' 9 RETURN = TERMINATE Program.'/)
      READ(*,9998) A
9998    FORMAT(I1)
```

```
      IF (A.EQ.9) GOTO 9999
      IF (A.EQ.1) GOTO 1310
      IF (A.EQ.6) GOTO 1006
9999  STOP
      END
```

## GLOSSARY OF TERMS AND ABBREVIATIONS

Lung volume and ventilation variables (liters BTPS):

ERV = expiratory reserve volume

FRC = functional residual capacity

IC = inspiratory capacity

IRV = inspiratory reserve volume

RV = residual volume

TLC = total lung capacity

VC = vital capacity

$V_T$ = tidal volume

$V_E$ = minute ventilation derived from expiratory data

MVV = maximal voluntary ventilation

$FEV_{1.0}$ = forced expired volume during first second of maximal expiration

$FEV_{1.0}/VC$ = ratio of $FEV_{1.0}$:vital capacity expressed as a percentage.

Respiratory flow variables (liters.second $^{-1}$):

PEFR = peak expiratory flow rate

$MEFR_{25}$ = maximum expiratory flowrate at 25% of VC

$MEFR_{50}$ = maximum expiratory flowrate at 50% of VC

$MEFR_{75}$ = maximum expiratory flowrate at 75% of VC.

Respiratory timing variables (seconds):

$T_I$ = time of inspiration

$T_E$ = time of expiration

$T_{TOT}$ = total respiratory duration

Post-exp pause = duration of the pause between the end of expiration and the commencement of the next inspiration

Post-insp pause = duration of the pause between the end of inspiration and the commencement of the next expiration

$T_I/T_{TOT}$ = ratio of the time of inspiration to the total cycle duration

$V_T/T_I$ = ratio of the tidal volume to the inspiratory duration ($l.s^{-1}$).